Date of acceptance Grade

Instructor: Arto Klami, PhD

Scalable Bayesian Induction of Word Embeddings

Joseph H. Sakaya

Helsinki November 3, 2015

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Joseph H. Sakaya

Työn nimi — Arbetets titel — Title

Scalable Bayesian Induction of Word Embeddings

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	November 3, 2015	72 pages + 0 appendices

Tiivistelmä — Referat — Abstract

Traditional natural language processing has been shown to have excessive reliance on human-annotated corpora. However, the recent successes of machine translation and speech recognition, ascribed to the effective use of the increasingly availability of web-scale data in the wild, has given momentum to a re-surging interest in attempting to model natural language with simple statistical models, such as the n-gram model, that are easily scaled. Indeed, words and word combinations provide all the representational machinery one needs for solving many natural language tasks.

The degree of semantic similarity between two words is a function of the similarity of the linguistic contexts in which they appear. Word representations are mathematical objects, often vectors, that capture syntactic and semantic properties of a word. This results in words that are semantic cognates having similar word representations, an important property that we will widely use. We claim that word representations provide a superb framework for unsupervised learning on unlabelled data by compactly representing the distributional properties of words.

The current state-of-the-art word representation adopts the skip-gram model to train shallow neural networks and presents negative sampling, an idea borrowed from Noise Contrastive Estimation, as an efficient method of inducing embeddings. An alternative approach contends that the inherent multi-contextual nature of words entails a more Canonical Correlation Analysis-like approach for best results. In this thesis we develop the first fully Bayesian model to induce word embeddings. The prominent contributions of this thesis are:

1. A crystallisation of the best practices from previous literature on word embeddings and matrix factorisation into a single hierarchical Bayesian model.
2. A scalable matrix factorisation technique for structured sparse data.
3. Representation of the latent dimensions as continuous Gaussian densities instead of as point estimates.

We analyse a corpus of 170 million tokens and learn for each word form a vectorial representation based on the 8 surrounding context words with a negative sampling rate of 2 per token. We would like to stress that while we certainly hope to beat the state-of-the-art, our primary goal is to develop a *stochastic* and *scalable* Bayesian model. We evaluate the quality of the word embeddings against the word analogy tasks as well as other such tasks as word similarity and chunking. We demonstrate competitive performance on standard benchmarks.

ACM Computing Classification System:
Computing methodologies∼Machine learning
Computing methodologies∼Natural language processing
Mathematics of computing∼Variational methods

Avainsanat — Nyckelord — Keywords
Bayesian, word embeddings, induction, stochastic, scalable

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

> I have a little brown cocoon of an idea that may possibly expand into a magnificent moth of fulfilment.
>
> *Anne's House of Dreams*

These last few years have seen a wide flourish of successful attempts at solving natural language processing tasks such as machine translation [Koe10, BPPM93] and speech recognition [RS10, RJ93] and more sedate forays into others such as question answering. A good part of this success is owed to machine learning algorithms – mostly shallow, statistical models – that despite blithely ignoring the linguistic intricacies of the data they attempt to model, perform surprisingly well on standard benchmarks. This has given momentum to a re-surging interest in attempting to model natural language using simple statistical models. While we strongly advocate the use of machine learning to aid progress in natural language tasks, we also qualify such an espousal by noting that most progress in natural language is limited to specialised domains and a generic framework that allows for a computer to understand natural language has yet to be developed. Until such a time, we are content to incrementally develop unsupervised algorithms that tackle niches.

The demonstrable efficacy of such unsupervised algorithms when used with copious amounts of data has slowly begun edging out earlier supervised techniques that were as much a product of human endeavour as of computers into obsolescence [SACJ11]. As Halevy *et al* [HNP09] have rightly pointed out, the success of machine translation can be ascribed not so much to the increased complexity of the algorithms used as to the existence of massive training sets of input–output behaviour that comes to us *in the wild*, in contrast to more traditional natural language problems such as parsing and part-of-speech tagging that heavily rely on human-annotated corpora to achieve satisfactory results. Therefore, instead of constructing algorithms with the eventual hope of stumbling upon annotated data, we reckon it more congenial to develop such ones as learn from *existing* unlabelled data[1]. We also wish to make it amply clear that we use unlabelled data in the order of gigabytes, if only because it exists, and that it would be daft not to avail oneself of the free web-scale data to

[1] We duly note that the much-vaunted deep learning algorithms shown to be state-of-the-art in machine vision and speech recognition owe their success not to unlabelled data but to copious amounts of *labelled* data.

improve our model parameters. We do *not* advocate *big data* as an end in itself.

The crux of unsupervised learning is to design a model described by a set of parameters $\boldsymbol{\Theta}$ that *adequately* – the precise definition of which varies from problem to problem – reflects the reality we are faced with. Given a set of data $\mathcal{D}_N$ of N points, one infers an optimal $\boldsymbol{\Theta}^*$ such that

$$\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} p(\mathcal{D}_N|\boldsymbol{\Theta}).$$

This is called *maximum likelihood estimation* (MLE). However, the model is too simplistic as it (a) does not allow for any encoding prior knowledge about the model parameters; and, (b) provides only point estimates. Point (a) is usually resolved by introducing priors over $\boldsymbol{\Theta}$:

$$\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} \frac{p(\mathcal{D}_N|\boldsymbol{\Theta})p(\boldsymbol{\Theta})}{p(\mathcal{D}_N)}.$$

This type of inference is known as *maximum a posteriori* (MAP). To resolve point (b) one opts for a more Bayesian approach which favours distributions that give a *credible range* over the posteriors, over point estimates. Thus, the posterior distribution is defined [BS94] to be

$$p(\boldsymbol{\Theta}^*|\mathcal{D}_N) = \frac{p(\mathcal{D}_N|\boldsymbol{\Theta})p(\boldsymbol{\Theta})}{\int p(\mathcal{D}_N|\boldsymbol{\Theta})p(\boldsymbol{\Theta})\, d\boldsymbol{\Theta}}.$$

It follows that that when N tends to 0 the posterior is the prior and as N tends to infinity the mean of the posterior would be centred on the MLE. Of course, the elegance of such an approach is shattered when we see that the denominator (known as the marginal likelihood) is in all but the simplest of formulations an intractable integral. In most practical cases integrating over a dimensionality of the order of millions is prohibitive. One therefore generally resorts to stochastic approximations such as Markov Chain Monte Carlo (MCMC), and repeatedly draws samples to inspect statistics. While MCMC theoretically promises asymptotically exact posteriors, we contend that such procedures are computationally prohibitive and in practice produce just finite time approximations. Instead, we use deterministic approximation techniques such as variational inference, an attractive proposition that is not only scalable, but also allows one to perform easy model selection by producing a variational lower bound as a spin-off. More about variational inference can be found in Chapter 2. Perhaps the strongest argument in favour of the Bayesian methodology is this: when $\boldsymbol{\Theta}$ consists of latent variables as well as model parameters, MLE and MAP require evaluation of the parameters even if our desideratum

were to find the distribution over the latent variables. Here, latent variables are hidden variables that are responsible for generating the observed data and are generally of a lower dimension than the observed data. For example, in the case of mixture models, the latent variable is the cluster assignment of the observed data point. If $\mathbf{Z}$ constitutes the latent parameters, then

$$p(\mathbf{Z}^*|\mathcal{D}_N) = \int p(\mathbf{Z}|\mathcal{D}_N, \mathbf{\Theta}) p(\mathbf{\Theta}|\mathcal{D}_N)\, d\mathbf{\Theta}.$$

The Bayesian approach thus allows more robust inference of the latent variables by marginalising over the model parameters [GG07].

1.1 Problem Definition

Words are known by the contexts they appear in. One can turn this contextual information to good account and infer semantic and syntactic properties of a word. Word representations are real-valued vectors that accurately capture these properties. We learn a k-dimensional vector for each word such that the vectors of word pairs that behave similarly in a language tend to have a smaller Euclidean distance between them than those that don't behave similarly. Consider, for example, the words *Finland* and *Sweden*. We see from Figure 1 that the contexts they appear in are largely similar and therefore expect these words to be either semantic or syntactic cognates and the vectors learnt for these words to have a small Euclidean distance. However, the contexts of any one of these words, say *Finland*, differs quite drastically from those of *stocks* as shown in Figure 1. The vectors learnt for *Finland* and *stocks* therefore have a very large Euclidean distance between them. One utilises the contexts of a word to *unsupervisedly* learn its vector representation. This results in words that have similar contexts having similar vector representations. This simply implies that similar words have smaller Euclidean distances amongst their vector representations. This property can be, for instance, used in retrieving all the synonyms of a given word.

Let $\mathcal{V}$ be the total number of unique words (the size of the vocabulary) in the corpus $\mathcal{D}$. The input to the Bayesian model is given as running text, where each word and its left and right contexts are hashed into an integer between 0 and $|\mathcal{V}|$, succinctly representing a sparse binary vector. A hashed word w and its hashed contexts $c \in C(w)$ can thus be modelled as entries in a sparse vector of dimension $1 \times |\mathcal{V}|$. Such a vector $\mathbf{x}_w$ for w is set to one for all $c \in C(w)$ and to zero otherwise, known traditionally know in the word embeddings literature as the *one-hot* representation

Figure 1: The words appearing as contexts of *Finland* (top left) and *Sweden* (top right). Since the contexts of these two words overlap to a significant extent, we could argue that the *Finland* and *Sweden* share syntactic and semantic properties. However, the contexts of either of these words differs significantly from those of *stocks* (bottom). Thus, the vectors learnt for *Finland* and *Sweden* would have a small Euclidean distance with each other. The Euclidean distance between the vectors of either *Finland* or *Sweden* and *stocks* is much higher.

[TRB10]. Usually, all $c' \notin C(w)$ implicitly are assumed to be zero. For $|\mathcal{D}|$ such tokens, this constitutes a sparse binary matrix $\mathbf{X}$. It is this matrix that we seek to factorise as a product of $\mathbf{U}$ and $\mathbf{V}$.

In typical matrix factorisation problems the dimension of $\mathbf{U}$ is $|\mathcal{D}| \times k$ and that of $\mathbf{V}$ is $|\mathcal{V}| \times k$, where k is the dimension of the embedding. This provides each entry

in $\mathbf{X}$ with a corresponding unique latent entry in $\mathbf{U}$. However, we recognise that not all tokens of $|\mathcal{D}|$ are unique and that several of them correspond to a unique word that can be represented as a single latent entry in $\mathbf{U}$. To account for this, we instead use a $\mathbf{U}$ of dimension $|\mathcal{V}| \times k$. Thus every occurrence of $w \in \mathcal{D}$ is mapped to a unique $\mathbf{u}_w \in \mathbf{U}$. Furthermore, since the entries of $\mathbf{X}$ are sparse, we propose to use the logistic model for factorisation instead of the more typical Gaussian model. This affords us the flexibility to either actively penalise every $c' \notin C(w)$ or simply ignore those $c' \notin C(w)$ that could potentially appear in a later token of the same word. Having defined our model, we employ variational inference to induce the actual word embeddings over the corpus. To make our model scalable, we use two variants: firstly, we employ stochastic variational inference; secondly, we use a multithreaded implementation. This results in a latent embedding $\mathbf{u}_w$ for every w that accounts for all possible contexts w appears in.

[CWB+11] first showed how these word representations come in handy in augmenting such tasks as chunking, semantic role labelling and named entity recognition. Since then [DRFU12], [MCCD13], [LL13] and [LC14] have improved upon these results to furnish better and better embeddings. In this thesis, the Bayesian model we present not only infers embeddings with good semantic properties but is also the first to offer continuous Gaussian densities as part of its inference process.

1.2 Contributions

This thesis presents a matrix factorisation model that attempts to model word-context tuples as lower-dimensional vectors with high predictive accuracy. Such vectors can be used to bolster existing NLP systems and push the state-of-the-art. The following contributions feature prominently in this work:

1. A crystallisation of the best practises from previous literature on word embeddings and matrix factorisation into a single hierarchical Bayesian model.

2. We derive and implement a scalable, stochastic binary matrix factorisation model for huge sparse binary matrices. Such a model finds use not only in inducing word embeddings, but also in movie recommendation tasks and document subspace clustering [ZLDZ07].

3. Representation of the latent dimensions as continuous Gaussian densities instead of as point estimates.

1.3 Layout

Chapter 2 provides a brief mathematical background. Chapter 3 on word representations gives a brief review of a formal definition of vector space representation as well as a host of previous works done on it. Chapter 4 develops a Bayesian matrix factorisation model that will be used in learning word embeddings. Chapter 5 will list a few practical tweaks that we add to make the inference in the Bayesian model scalable and practical. We present our results in Chapter 6. Chapter 7 draws conclusions and briefly retrospects our various claims, while Chapter 8 enumerates a few possibilities that could be adopted as extensions to the existing model.

2 Mathematical Background

> But if you call me Anne, please call me Anne with an 'e'
>
> *Anne of Green Gables*

We saw previously how we run into difficulties in Bayesian inference during estimation of $p(\mathbf{Z}|\mathcal{D})$ owing to its intractability. One solution is to use an Expectation-Maximisation style algorithm, by calculating the log likelihood expectation of the data over the posterior distributions of the parameters. However we run into one of the following problems [Bis06]:

1. The dimensionality of the latent space is too high to work with, prohibiting numerical integration.
2. The complexity of the posterior distribution renders computing expectations analytically intractable with no closed-form solutions.

Variational approximation provides a deterministic framework by allowing us to approximate the posterior by

1. Allowing the distributions to have parametric forms, such as the exponential family, which in addition to being summarised by sufficient statistics have conjugate priors.
2. Allowing specific factorisations over the posterior (called mean field factorisation [Par88]).

In this chapter we briefly introduce the exponential family and how it relates to the variational scheme.

2.1 Exponential Family

A member of the exponential family of distributions over $\mathbf{x}$ can be represented as

$$p(\mathbf{x}|\boldsymbol{\lambda}) = h(\mathbf{x}) \exp(\boldsymbol{\lambda}^T \cdot T(\mathbf{x}) - A(\boldsymbol{\lambda})).$$

Distribution	$\boldsymbol{\lambda}$	$T(\mathbf{x})$	$A(\boldsymbol{\lambda})$	$h(\mathbf{x})$
Gaussian	$\begin{bmatrix} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ -\frac{1}{2}\boldsymbol{\Sigma}^{-1} \end{bmatrix}$	$\begin{bmatrix} \mathbf{x} \\ \mathbf{x}\mathbf{x}^{\mathrm{T}} \end{bmatrix}$	$\frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \frac{1}{2}\ln\lvert\boldsymbol{\Sigma}\rvert$	$(2\pi)^{-\frac{k}{2}}$
Gamma	$\begin{bmatrix} \alpha - 1 \\ -\beta \end{bmatrix}$	$\begin{bmatrix} \ln x \\ x \end{bmatrix}$	$\ln\Gamma(\alpha) - \alpha\ln\beta$	1
Dirichlet	$\begin{bmatrix} \alpha_1 - 1 \\ \vdots \\ \alpha_k - 1 \end{bmatrix}$	$\begin{bmatrix} \ln x_1 \\ \vdots \\ \ln x_k \end{bmatrix}$	$\sum_{i=1}^{k}\ln\Gamma(\alpha_i) - \ln\Gamma\left(\sum_{i=1}^{k}\alpha_i\right)$	1
Categorical	$\begin{bmatrix} \ln p_1 \\ \vdots \\ \ln p_k \end{bmatrix}$	$\begin{bmatrix} [x=1] \\ \vdots \\ [x=k] \end{bmatrix}$	0	1

Table 1: Some common distributions represented as exponential family distributions.

Here,

$$\begin{aligned} \boldsymbol{\lambda} &\quad \text{constitutes the natural parameter,} \\ T(\mathbf{x}) &\quad \text{constitutes the sufficient statistics,} \\ A(\boldsymbol{\lambda}) &\quad \text{constitutes the log normaliser,} \\ h(\mathbf{x}) &\quad \text{constitutes the base measure.} \end{aligned}$$

Note that $\mathbf{x}$ could be discrete or continuous, scalar or vector. In fact, most familiar distributions can be recast as a member of the exponential family. Some common examples are shown in Table 1.

2.2 Conjugate priors

The usefulness of the exponential family becomes more apparent when we realise that each of its members has a conjugate prior which can be represented as

$$p(\boldsymbol{\lambda}|\boldsymbol{\chi},\nu) = f(\boldsymbol{\chi},\nu)\exp(\boldsymbol{\lambda}^{\top}\boldsymbol{\chi} - \nu A(\boldsymbol{\lambda}))$$

where ν is the number of observations to which the prior contributes, $\boldsymbol{\chi}$ is the number of pseudo-observations contributing to the sufficient statistics and $f(\boldsymbol{\chi},\nu)$

is a normalisation constant. $\nu A(\boldsymbol{\chi})$ is the log-normaliser and can also be re-written as

$$p(\boldsymbol{\lambda}|\boldsymbol{\chi},\nu) = f(\boldsymbol{\chi},\nu)g(\boldsymbol{\lambda})^{\nu}\exp(\boldsymbol{\lambda}^{\top}\boldsymbol{\chi}).$$

Consider now the data set $\mathcal{D} = \{\mathbf{d}_1 \dots \mathbf{d}_n\}$ where

$$\begin{aligned} p(\mathcal{D}|\boldsymbol{\lambda}) &= \prod_i p(\mathbf{d_i}|\boldsymbol{\lambda}) \\ &= \prod_i h(\mathbf{d_i})g(\boldsymbol{\lambda})\exp(\boldsymbol{\lambda}^T \cdot T(\mathbf{d_i})) \\ &= \prod_i h(\mathbf{d_i})g(\boldsymbol{\lambda})^n \exp(\boldsymbol{\lambda}^T \cdot \sum_i^n T(\mathbf{d_i})). \end{aligned}$$

The posterior can thus be defined as

$$\begin{aligned} p(\boldsymbol{\lambda}|\mathcal{D},\boldsymbol{\chi},\nu) &\propto p(\mathcal{D}|\boldsymbol{\lambda})p(\boldsymbol{\lambda}|\boldsymbol{\chi},\nu) \\ &= g(\boldsymbol{\lambda})^n \exp(\boldsymbol{\lambda}^T \cdot \sum_i^n T(\mathbf{d_i})) \cdot g(\boldsymbol{\lambda})^{\nu}\exp(\boldsymbol{\lambda}^{\top}\boldsymbol{\chi}) + \text{const} \\ &= g(\boldsymbol{\lambda})^{\nu+n}\exp(\boldsymbol{\lambda}^{\top}\boldsymbol{\chi} + \boldsymbol{\lambda}^T \cdot \sum_i^n T(\mathbf{d_i})) + \text{const}. \end{aligned}$$

Owing to conjugacy the posterior has the same functional form as the prior and is defined as

$$p(\boldsymbol{\lambda}|\mathcal{D},\boldsymbol{\chi},\nu) = p(\boldsymbol{\lambda}|\boldsymbol{\chi} + \sum_i^n T(\mathbf{d_i}), \nu + n).$$

The sufficient statistics alone are enough to infer the posterior of $\boldsymbol{\lambda}$.

2.3 Variational Approximation

Consider the marginal likelihood $p(\mathcal{D})$ of the data set $\mathcal{D}$ which is also intractable. That is,

$$p(\mathcal{D}) = \int p(\mathcal{D},\theta)\, d\theta$$

where θ includes the parameters as well as the latent variables and are more often than not vector-valued. The variational approximation [Att99] introduces a parametric distribution q to approximate true posterior:

$$\begin{aligned} \ln p(\mathcal{D}) &= \ln \int q(\theta)\frac{p(\mathcal{D},\theta)}{q(\theta)}\, d\theta \\ &\geq \int q(\theta)\ln\frac{p(\mathcal{D},\theta)}{q(\theta)}\, d\theta \\ &= \mathcal{L}(q). \end{aligned}$$

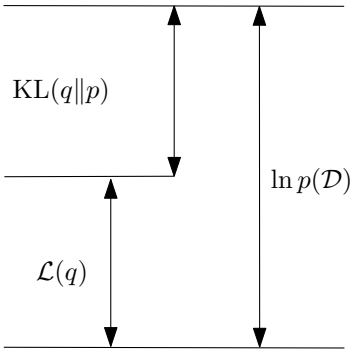


Figure 2: Variational Bayes in a nutshell [Bis06]: $\mathcal{L}(q)$ lower bounds the log evidence $\ln p(\mathcal{D})$. To achieve the best bound, we maximise $\mathcal{L}(q)$ over q, which effectively minimises $\mathrm{KL}(q\|p)$ letting q approximate p better.

The KL divergence between the two continuous distributions q and p is given as [Bis06]

$$\mathrm{KL}(q\|p) = \int q(\theta) \ln \frac{q(\theta)}{p(\theta|\mathcal{D})} \,\mathrm{d}\theta$$

and the expected lower bound $\mathcal{L}(q)$ as,

$$\mathcal{L}(q) = -\int q(\theta) \ln \frac{q(\theta)}{p(\theta, \mathcal{D})} \,\mathrm{d}\theta.$$

This leads to the following equivalence between the KL divergence and the marginal likelihood:

$$\begin{aligned}
\mathrm{KL}(q\|p) &= \int q(\theta) \ln \frac{q(\theta)}{p(\theta|\mathcal{D})} \,\mathrm{d}\theta \\
&= \int q(\theta) \ln \frac{q(\theta)}{p(\mathcal{D}, \theta)} \,\mathrm{d}\theta + \ln p(\mathcal{D}) \\
&= -\mathcal{L}(q) + \ln p(\mathcal{D}).
\end{aligned}$$

The relationship between $\mathcal{L}(q)$, $\mathrm{KL}(q\|p)$ and $\ln p(\mathcal{D})$ is given in Figure 2. As KL is non-negative,

$$\begin{aligned}
\ln p(\mathcal{D}) &\geq -\int q(\theta) \ln \frac{q(\theta)}{p(\mathcal{D}, \theta)} \,\mathrm{d}\theta \\
&= \mathrm{E}_q[p(x, \theta)] - \mathrm{E}_q[q(\theta)] \\
&= \mathcal{L}(q).
\end{aligned}$$

In other words, $\mathcal{L}(q)$ lower bounds the marginal log likelihood. To achieve the best bound, we maximise $\mathcal{L}(q)$ over q, which effectively minimises the $\mathrm{KL}(q\|p)$. Of course, we see that $\mathcal{L}(q)$ is intractable as well. To evaluate $\mathcal{L}(q)$, we will assume a factorised form of $q(\theta)$, to wit, $q(\theta) = \prod_i q(\theta_i)$ where θ_i are the parameters we need to optimise over. We then do a gradient-based optimisation for each $q(\theta_i)$,

$$\nabla_{\lambda_l}\mathcal{L} = \nabla_{\lambda_l}\mathbb{E}_q[\ln p(\mathcal{D},\theta)] - \nabla_{\lambda_l}\mathbb{E}_q[\ln q(\theta|\lambda)].$$

This so-called mean field factorisation allows one to factorise $q(\theta|\lambda)$ as, $q(\theta|\lambda) = \prod_i q_i(\theta_i|\lambda_i)$. Let us suppose each of $q(\theta_i|\lambda_i)$ is a member of the exponential family. Then, they have the following parametrisation:

$$\begin{aligned} q_i(\theta_i|\lambda_i) &= h(\theta_i)\exp(\lambda_i^\top t(\theta_i) - A(\lambda_i)) \\ &= h(\theta_i)\exp(\lambda_i^\top \frac{\partial A(\lambda_i)}{\partial \lambda_i} - A(\lambda_i)). \end{aligned}$$

This leads to,

$$\begin{aligned} \nabla_{\lambda_l}\mathrm{E}_q[\ln q(\theta_i|\lambda_i)] &= \frac{\partial}{\partial\lambda_i}(\lambda_i^\top \frac{\partial A(\lambda_i)}{\partial\lambda_i} - A(\lambda_i)) \\ &= \lambda_i \frac{\partial^2 A(\lambda_i)}{\partial\lambda_i \partial\lambda_i^\top} \\ &= \lambda_i \mathcal{V}(\lambda_i). \end{aligned}$$

Similarly for $p(\mathcal{D},\theta)$, if a is the Markov blanket of θ_i – constituting its parents, its children and its children's other parents – and $S_a = \mathbb{E}_q(\ln p_a(\mathcal{D},\theta))$, each in the conjugate exponential family

$$\frac{\partial S_a}{\partial \lambda_i} = \mathcal{V}(\lambda_i)\mathbb{E}_q[\eta_a(x,\theta_{-i})]$$

where $\eta_a(x,\theta_{-i})$ constitutes the natural parameter. Thus,

$$\begin{aligned} \nabla_{\lambda_l}\mathcal{L} &= \mathcal{V}(\lambda_i)\mathbb{E}_q[\eta_a(x,\theta_{-i})] - \mathcal{V}(\lambda_i)\lambda_i \\ &= \mathcal{V}(\lambda_i)(\mathbb{E}_q[\eta_a(x,\theta_{-i})] - \lambda_i) \\ &= 0 \\ \implies \lambda_i &= \sum_a \mathbb{E}_q(\eta_a(x,\theta_{-i})). \end{aligned}$$

In summary, we can write the update equations tersely as follows

$$q(\theta_i|\lambda_i) \propto \exp(\mathbb{E}_{q_{-i}}[\ln p(\mathcal{D},\theta)])$$

Algorithm 1: Mean Field Variational Bayes.

input : The data set $\mathcal{D}$ and appropriately parametrised $q(\theta_i|\lambda_i)\ \forall i \in [1, w]$

output: $\lambda_i,\ \forall i$

```
while evidence lower bound ℒ not converged do
    for i ← 1 to w do
        q(θ_i|λ_i) ∝ exp(𝔼_{q_{−i}}[ln p(𝒟, θ)])
```

leading to the alternate maximisation scheme presented in Algorithm 1.

In this chapter we presented a deterministic framework to approximate the posterior $p(\theta|\mathcal{D})$ by $q(\theta)$ factorised as $q(\theta) = \prod_i q(\theta_i|\lambda_i)$ where each $q(\theta_i)$ belongs to the conjugate exponential family parametrised by λ_i with the posteriors retaining the same distribution as the prior. We then resort to alternative optimisation techniques such as in Algorithm 1 to iteratively estimate the values of λ_i that minimise the expected lower bound. We will in later sections introduce more efficient optimisation algorithms that scale well over large data sets.

3 Word Representations

> "Words aren't made – they grow"
> said Anne.
>
> *Anne of the Island*

Noscitur a sociis or the notion that a word is known by the company it keeps, is probably as old and intuitive a concept as language itself – we often catch ourselves teasing out a word's meaning with a little hand from its neighbours. This was first formalised in linguistics by Zellig Harris[2], when he stated that *"The degree of semantic similarity between two linguistic expressions A and B is a function of the similarity of the linguistic contexts in which A and B can appear"*.

It comes as no surprise then that this notion can be extended to draw plausible conclusions about the semantic nature of words. Simply put, words occurring in similar contexts are semantic cognates. Such relations have a broad and sweeping coverage in that they accommodate anything from synonyms and antonyms to hypernyms and hyponyms within their purview.

Luckily, this intuition is easily translated into simple probabilistic models that adequately capture the semantic properties. The resulting representations drawn are inherently valuable as they encode the predictive nature of a word as a vector. However, their real worth is manifest in their power to push the current state-of-the-art systems – essentially improvements that cost nothing in terms of human labour[3].

As an illustration, consider the words *Harvard*, *Stanford* and *Princeton*. We intuit that since these all are private universities in the United States, they are bound to have similar neighbours in a good part of their occurrences. It wouldn't be unreasonable to expect words like *graduate*, *school* or *university* to occur in their vicinity. We list down for the benefit of the reader the top occurrences of each of these words. If the word representations are accurate at all, this triad should have similar representations. We will later see in Chapter 6 if our rationale is justified.

[2]We wryly observe that much of Harris' work is absent from mainstream linguistics. Known to cite only himself and his colleagues, he has had the courtesy largely returned. He perhaps owes it to the machine learning community for breathing new life into his work.

[3]That is, if one were to discount the initial effort spent on putting such a system in place.

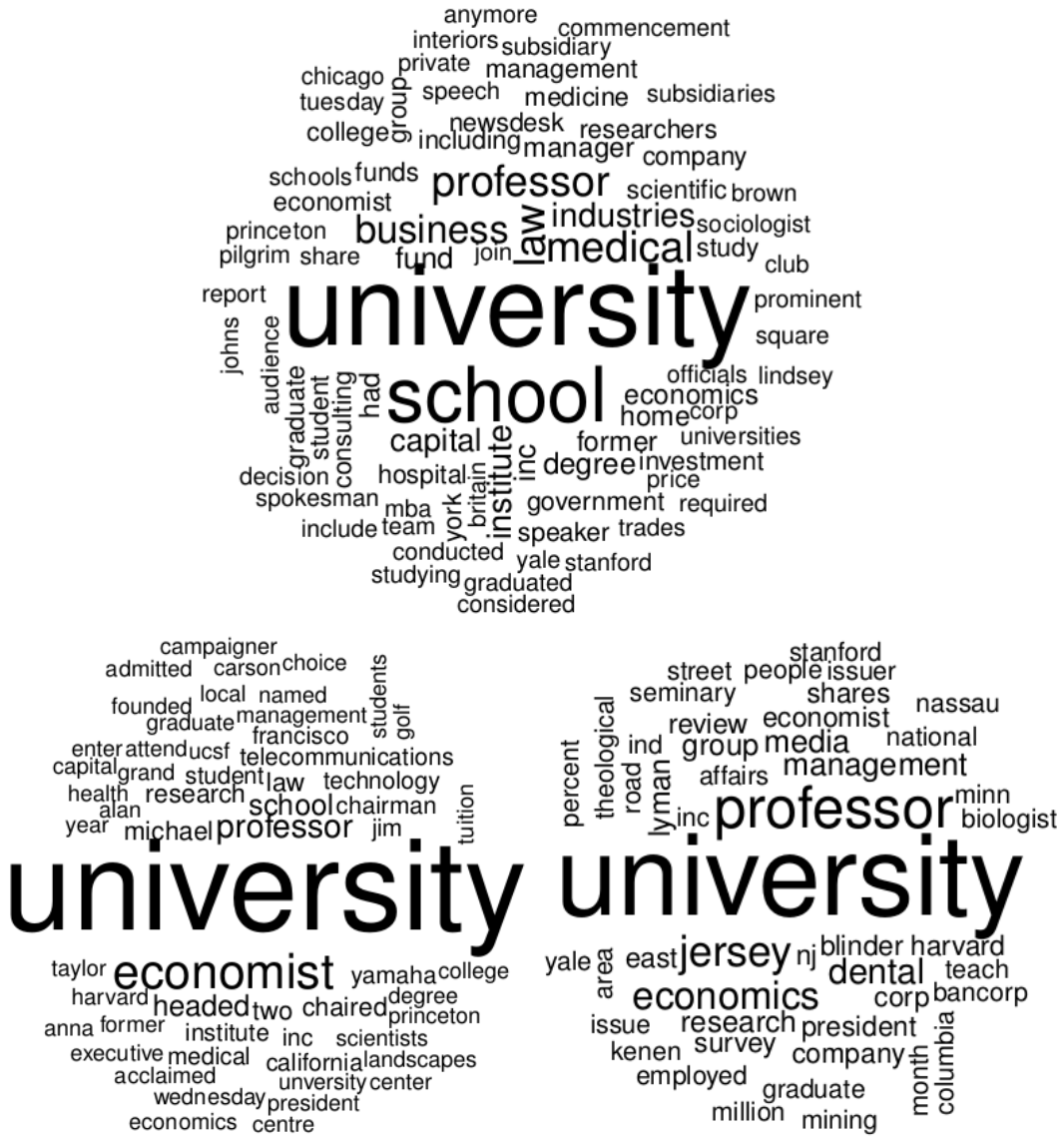


Figure 3: Contextual word clouds for 607 occurrences of *Harvard*, 367 occurrences of *Stanford* and 270 occurrences of *Princeton*. The context window size is 2 to the left and right. The data has been cleaned to remove stop words. That *university* and *professor* figure so prominently in each cloud validates our claim of the three words having similar contextual distributions.

3.1 Definition

Formally, a *word representation* is a vector space model, with each dimension having either an explicit or an inherent grammatical or semantic interpretation. The similarity of two such vectors is given, say, by the cosine distance. The simplest form of such a word vector is the *one-hot* representation with a vector the size of

the vocabulary with a single dimension turned on. At the risk of stating the obvious, the sparsity and the high dimensionality result in poor parameter estimation. Our desideratum is therefore, a vector of reduced dimension with the discriminative power of the higher dimensional representation.

Consider a set $\mathcal{D}$ of word, context pairs (w, c) of size n with w drawn from a vocabulary of size d. Let $C(w)$ be the set of all contexts for a given w. We model $\mathcal{D}$ as a sparse binary matrix $\mathbf{X}$ using the one-hot method described above. That is, for each row entry w, only $c \in C(w)$ is turned on. The rest are assumed to be zero. This naturally yields a rather unwieldy matrix of dimensions $n \times d$. However, we obtain a reduced dimensional approximation by factorising $\mathbf{X} = \mathbf{UV}$. Usually such factorisation is carried out by MAP estimates which are prone to overfitting [SM08]. Alternatively, one could probabilistically cast $\mathbf{X}$ as a linear model with Gaussian noise [SM08]. That is, $\mathbf{x}_i \sim \mathcal{N}(\mathbf{u}_i^\top \mathbf{V}, \tau \mathbf{I})$. While [SM08] suggested a Gibbs sampler for inference, another option is to employ variational methods elaborated in Section 2.3. The resultant lower dimensional $\mathbf{U}$ captures several semantic and syntactic properties of the words.

We shall now discuss different kinds of word representations in literature and their limitations.

3.2 Global Representations

Distributional representations rely on the global co-occurrence matrices of words, where each row corresponds to the co-occurrence frequencies of a word with all its contexts, to generate reduced dimensional representations. This includes the widely popular Latent Semantic Analysis (LSA). LSA [LD97] works by performing *Singular Value Decomposition* (SVD) on the global co-occurrence matrix $\mathbf{X} = \mathbf{U\Sigma V}^\top$ where $\mathbf{U}$ is an orthogonal matrix of the eigenvectors of $\mathbf{XX}^\top$, $\mathbf{V}$ is an orthogonal matrix of the eigenvectors of $\mathbf{X}^\top\mathbf{X}$, and $\mathbf{\Sigma}$ is diagonal matrix with the square roots of the non-zero eigenvalues of $\mathbf{XX}^\top$ as its diagonals. Choosing the k largest eigenvalues corresponds to a k-dimensional reduction, which is the best least square approximation to the original matrix $\mathbf{X}$.

[Kas98] introduced a random-mapping based dimensionality reduction $\mathbf{Z} = \mathbf{RX}^\top$ where $\mathbf{R}$ is a random matrix with normalised columns and $\mathbf{Z}$ is the matrix with reduced dimensionality d. Given $\mathbf{a}', \mathbf{b}' \in \mathbf{Z}$ and $\mathbf{a}, \mathbf{b} \in \mathbf{X}$, we see that $\mathbf{a}'^\top\mathbf{b}' = \mathbf{a}^\top\mathbf{R}^\top\mathbf{Rb}$. Representing the matrix $\mathbf{R}^\top\mathbf{R}$ as $\mathbf{I} + \epsilon_{i,j}$ where $\epsilon_{i,i} = 0$ and $\epsilon_{i,j} = \mathbf{r}_i\mathbf{r}_j$,

[Kas98] proved that $\mathrm{E}(\epsilon_{i,j}) = 0$ and $\sigma_\epsilon^2 \approx \frac{1}{d}$. Clearly, the higher the value of d, the better the approximation is. More specifically, if $\mathbf{a'}^\top \mathbf{b'} = \mathbf{a}^\top \mathbf{b} + \delta$, where $\delta = \sum_{k \neq l} \epsilon_{k,l} a_k b_l$, it was proved that $\mathrm{E}(\delta) = 0$ and $\sigma_\delta^2 \leq 2\sigma_\epsilon^2 \approx \frac{2}{d}$. That is, the distortion of $\mathbf{a'}^\top \mathbf{b'}$ has zero mean and a variance of at most $\frac{2}{d}$.

[LB96]'s Hyperspace Analogue to Language (HAL) select columns with the highest variances from the co occurrence matrix $\mathbf{X}$ as the latent dimensions while [VH05] uses Independent Component Analysis (ICA) to transform $\mathbf{X}$ to a lower dimension. However, distributional techniques have proved to be expensive in general, and [TRB10] claims that the exact settings necessary for a distributional representation to be used for prediction tasks such as parsing and for labelling tasks such as chunking is not well understood.

3.3 Cluster-based Representations

As early as 1992, Brown et al [BdM$^+$92] in their seminal paper attempted to build a coherent n-gram based language model that was capable of predicting a word from the previous n words and developed strategies to assign these words to classes based on the frequency of their co-occurrence with other words. Suppose $f : V \mapsto C$ is a function that maps each word $w \in V$ to a word class $c \in C$, where $|C| << |V|$. One then defines a n-gram class based model as

$$p(w_k|w_1^{k-1}) = p(w_k|c_k)p(c_k|c_1^{k-1}) \quad \forall k \in [1, n].$$

For $k = 2$, [BdM$^+$92] defined the quality of such a mapping f as

$$\begin{aligned} L(f) &= \frac{1}{n} \ln \prod p(w_i|c_i)p(c_i|c_{i-1}) \\ &= \frac{1}{n} \sum_{w,w'} \ln p(w|c)p(c|c') \\ &= \sum_w p(w) \ln p(w) + \sum_{c,c'} p(c,c') \ln \frac{p(c,c')}{p(c)p(c')} \\ &= -H(w) + I(c,c'). \end{aligned}$$

where $H(w)$ is the entropy of the unigram word distribution and $I(c, c')$ is the average mutual information of the adjacent classes. [Lia05] presents an optimised clustering algorithm defined in terms of $I(c, c')$. A disadvantage of [BdM$^+$92] is that it is limited to bigram contexts alone and has to be redefined for larger context windows. This largely cripples the scalability of the bigram model, which is perhaps its only selling point.

3.4 Distributional Representations

In contrast to earlier representations, distributed representations rely on their immediate contexts to generate dense, low-dimensional and real-valued representations. Collobert et al. [CWB+11] in their phenomenal paper presented a unified neural network model to demonstrate that such word embeddings could be used for regular NLP tasks such as chunking, semantic role labelling and named entity recognition. While it was impressive as a demonstration of the use of word embeddings, it performed abysmally in comparison to later work, both in terms of predictive accuracy and training time.[4] [MH07] presented a log-bilinear algorithm that learnt a neural network that predicted the embeddings of the n^{th} word given the embeddings of the previous $n-1$ words.

3.4.1 word2vec

If $c \in C(w)$ is the context for $w \in V$, then the skip-gram model is defined by the objective function:

$$\arg\max_{v_c\, v_w} \sum_{w \in \mathcal{C}} \sum_{c \in C(w)} \ln p(c|w) \qquad p(c|w) = \frac{\exp(\mathbf{v}_c^\top \mathbf{v}_w)}{\sum_{c \in \mathcal{C}} \exp(\mathbf{v}_c^\top \mathbf{v}_w)}.$$

In order to maximise the predictive power of a word, we take the gradient of $\ln p(c|w)$. However, computing $\nabla \ln p(c|w)$ is rendered impractical by the summation term in the denominator. If, instead, one replaces the softmax function with a logistic function, $\sigma(x) = (1 + \exp(x))^{-1}$ the objective becomes

$$\arg\max_{\mathbf{v}_c\, \mathbf{v}_w} \sum_{w \in \mathcal{C}} \sum_{c \in C(w)} \sigma(\mathbf{v}_c^\top \mathbf{v}_w)$$

which is trivially maximised by setting $\mathbf{v}_w^\top \mathbf{v}_c$ to arbitrarily large values.

[MCCD13] uses the skip-gram model and presents negative sampling, an idea borrowed from [GH12] as an efficient way to induce word embeddings. In order to discourage trivial solutions, one introduces (c, w) samples that do not exist in the corpus (hence, negative sampling) modifying the objective to

$$\arg\max_{\mathbf{v}_c\, \mathbf{v}_w} \left[\sum_{w \in \mathcal{C},\ c \in C(w)} \sigma(\mathbf{v}_c^\top \mathbf{v}_w) + \sum_{w \in \mathcal{C},\ c' \notin C(w)} \sigma(-\mathbf{v}_{c'}^\top \mathbf{v}_w) \right].$$

[4] [LL13], as a sane afterthought to their neural network architecture, proposed using Hellinger distance based PCA on the co-occurrence matrix of words. Since the word distributions are inherently discrete, metrics such as the Bhattacharya and the Hellinger distance are well suited for measuring the divergence between them.

It has been shown empirically that setting the noisy distribution to $\frac{3}{4}^{\text{th}}$ of the unigram probabilities of the word works reasonably well. Recently, [GL14] proved that the embeddings induced by [MCCD13] constitute implicit matrix factorisation, where the matrix to factorise consists of as its entries the point-wise mutual information between a word and its context.

3.4.2 Multi-view embeddings

[DRFU12] proposed using Canonical Correlation Analysis (CCA), a cousin of Principle Component Analysis (PCA), for pairs of matrices. For any two matrices $\mathbf{L}$ and $\mathbf{R}$, CCA finds a pair of linear projections $\mathbf{\Phi_L}$ and $\mathbf{\Phi_R}$, such that the correlation between $\mathbf{L\Phi_L}$ and $\mathbf{R\Phi_R}$ is maximised. This can be formally represented as

$$\max \frac{\mathrm{E}[\langle \mathbf{L}, \mathbf{\Phi_L} \rangle \langle \mathbf{R}, \mathbf{\Phi_R} \rangle]}{\sqrt{\mathrm{E}[\langle \mathbf{L}, \mathbf{\Phi_L} \rangle^2] \mathrm{E}[\langle \mathbf{R}, \mathbf{\Phi_R} \rangle^2]}}.$$

[DRFU12] claim that whilst assuming the n-gram model for natural language, the two left and right contexts are conditionally independent given the current word, forming the two views of the CCA. They posit that by exploiting the multi-view nature of such data, the resultant dimensionality reduction retains much of its predictive power. If $\mathbf{L}$ were to denote the left context matrix, $\mathbf{R}$ the right context, $\mathbf{C} = [\mathbf{L}\,\mathbf{R}]$ the entire context, and $\mathbf{W}$ the word matrix, the *one-step* CCA is formulated as

$$\mathbf{CCA}(\mathbf{W}, \mathbf{C}) \implies (\mathbf{A}, \mathbf{\Phi}_C)$$

and *two-step* CCA is defined as

$$\begin{aligned} \mathbf{CCA}(\mathbf{L}, \mathbf{R}) &\implies (\mathbf{\Phi}_R, \mathbf{\Phi}_R), \\ \mathbf{S} &\impliedby [\mathbf{L\Phi}_L, \mathbf{R\Phi}_R], \\ \mathbf{CCA}(\mathbf{S}, \mathbf{W}) &\implies (\mathbf{\Phi}_S, \mathbf{\Phi}_W). \end{aligned}$$

[DRFU12] provides further multi-view models which we shall not discuss as they are simply modified versions of CCA with randomised SVD as a preprocessing step.

We contend that the efficacy of CCA is not so much because of its taking into account the multi-view nature of the data as of its proper factorisation of its contextual matrices. In fact, in practice we have found that the left and right view are distributionally similar and there is no significant gain in viewing them as two separate, independent entities. Our contention is not entirely without rationale as the Hellinger PCA[LL13] has produced evaluation scores much better and faster than

the CCA algorithm. Rather than use negative sampling, they inadvertently model the missing contexts rather than the contexts themselves. Since they do it so perfectly and since the latent features correspond to the tokens themselves rather than the unique vocabulary items, falsely modelling missing data as negative instances does not affect their model adversely.

3.5 A Final Word

Before we begin and define our model, we make some observations and examine a few deficiencies in the extant ones:

1. We note that all previous word embedding literature uses empirical means to set the value of k, the dimension of the latent embeddings.
2. Insofar as the current methods are concerned, they all *limit* themselves to a given (albeit large) data. None of them are designed to work on data streams. We propose a method that can learn on practically infinite data in addition to a scalable algorithm from large but finite data.
3. The performance of word embeddings requires significant fine-tuning. Every new publication has either been a claim of pushing the state-of-the-art, or oddly enough, a retraction of a previous result. Hence we strongly suspect that these are fine-tuned to a given data set and task for optimal performance. In the following chapters, while we certainly hope to beat or match a few such benchmarks, our primary goal is to build an algorithm that is *scalable* and *stochastic*
4. Finally, we see that none of the existing models are Bayesian. As far as we know, ours is the first practical, completely Bayesian model to induce word embeddings. It will also be the first such model to induce continuous Gaussian distributions on the latent features instead of producing point estimates. This comes at a steep price though – we will have to model the covariance matrix for each vocabulary item.

This concludes our discussion of the existing literature on word embeddings. In the next chapter we address the issue of designing a feasible Bayesian model for word embedding induction.

4 Factor Analysis of Sparse Data

> Everything that's worth having is some trouble...
>
> *Anne of Avonlea*

We had already described, albeit briefly, how the skip-gram model introduced by [MCCD13] implicitly constitutes matrix factorisation – an observation reinforced by promising view-specific CCA results obtained by [DRFU12]. In this chapter we shall formally introduce Bayesian methods for matrix factorisation and CCA and describe an efficient and scalable Bayesian model for factorising *extremely* sparse binary matrices.

4.1 Matrix Factorisation

Matrix factorisation refers to the decomposition of a matrix into a product of two or more matrices as shown in Figure 4, usually with a mind to capture the interaction between two entities in terms of a lower-dimensional latent features.

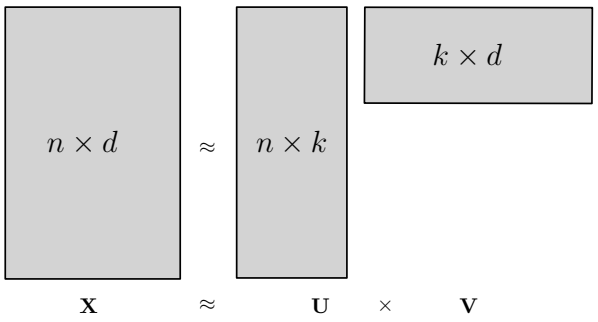


Figure 4: Matrix factorisation: The matrix $\mathbf{X}^{n \times d}$ with n rows each of dimension d, which is presumably high dimensional is factorised into two matrices $\mathbf{U}^{n \times k}$ and $\mathbf{V}^{k \times d}$ where $k << d$. In matrix factorisation literature the entries of $\mathbf{U}$ are known as the *latent features* or *factors* and those of $\mathbf{V}$ are known as *factor loadings*.

One of the most famous applications of matrix factorisations is arguably the recommender system[RV97]. For example, a large, sparse user-item matrix could be factorised into two smaller ones allowing us to determine relationships between users and between items or recommend items to users of similar interest[PKW14].

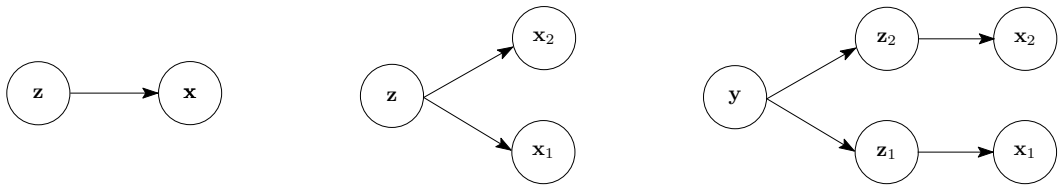


Figure 5: Graphical models for Bayesian factor analysis (left); CCA with shared latent variables (centre); CCA with view specific latent variables (right) – this is hardly ever used in practice.

4.2 Bayesian Factor Analysis

Matrix factorisation involves modelling correlations in higher-dimensional data as a lower-dimensional, oriented subspace [Bea03]. Consider a matrix $\mathbf{X} \in \Re^{d \times n}$ generated by a linear transformation of a zero-mean, unit-variance Gaussian matrix $\mathbf{U} \in \Re^{k \times n}$ and some additional zero mean diagonal covariance Gaussian noise $\boldsymbol{\epsilon}$, where $k \leq d$,

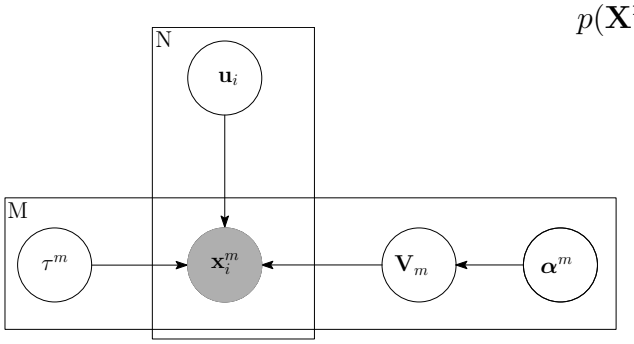
$$\mathbf{X} = \mathbf{U}^\top \mathbf{V} + \boldsymbol{\epsilon}$$

$$\mathbf{U}_i \sim \mathcal{N}(0, \mathbf{I}_k) \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \boldsymbol{\Psi}_d)$$

where $\mathbf{V} \in \Re^{k \times d}$ constitutes the linear transformation, also known as the *factor loadings* or the *projection* matrix. Marginalising over $\mathbf{U}_i$, it is not hard to see that $\mathbf{x}_i \sim \mathcal{N}(0, \mathbf{V}\mathbf{V}^\top + \boldsymbol{\Psi})$. By constraining the noise to be a diagonal covariance matrix, the $\mathbf{x}_i$ becomes conditionally independent for a given $\mathbf{u}_i$ [TB99]. This results in the $\boldsymbol{\epsilon}_i$ capturing the variation unique to an axis, while $\mathbf{V}$ captures the rest of the correlations presumed to be of interest. The components of the resulting latent vectors $\mathbf{U}_i$ are uncorrelated. Factor analysis differs from traditional PCA in that it differentiates between the variance and the correlations. This can be rectified by replacing Ψ with an isotropic covariance, in which case the model reduces to PCA. Incidentally, for a transformation such as $\mathbf{X}_i \to \mathbf{A}\mathbf{X}_i$ factor analysis is covariant under component-wise rescaling when $\mathbf{A}$ is diagonal, while Bayesian PCA is covariant under rotation when $\mathbf{A}$ is orthogonal [TB99].

4.3 Bayesian Canonical Correlation Analysis

CCA seeks to find linear components that capture correlations between two multivariate data sets. Given to two sets of random vectors $\mathbf{X}^1 \in \Re^{d_1 \times n}$ and $\mathbf{X}^2 \in \Re^{d_2 \times n}$ with a joint distribution, CCA first finds a pair of linear mappings to maximise



$$p(\mathbf{X^m}|\mathbf{U},\mathbf{V},\boldsymbol{\tau}) = \prod_{m,i} \mathcal{N}(\mathbf{x}_i^m|\mathbf{u}_i^T\mathbf{V}^m, \tau_m^{-1}\mathbf{I})$$
$$p(\mathbf{U}) = \prod_i \mathcal{N}(\mathbf{u}_i|0,\mathbf{I})$$
$$p(\boldsymbol{\tau}) = \prod_m \mathcal{G}(\tau_m|a,b)$$
$$p(\mathbf{V}|\boldsymbol{\alpha}) = \prod_{m,k} \mathcal{N}(\mathbf{V}_k^m|0,\alpha_k^m\mathbf{I})$$
$$p(\boldsymbol{\alpha}) = \prod_{m,k} \mathcal{G}(\alpha_k^m|a,b)$$

Figure 6: Plate diagram for the Bayesian CCA: Each entry in a given view $\mathbf{x}_i^m$ is modelled as a linear transformation of the latent vector $\mathbf{u}_i\mathbf{V}^m$ with some added noise. The dimensionality of $\mathbf{V}^m$ is automatically inferred. For CCA $M = 2$, while [VKKK14] presents presents a similar model for $M \geq 2$.

correlation between the two sets. This constitutes the first pair of new coordinates. The next pair of linear transformations maximises the correlation, conditional on it being uncorrelated to the first pair. [KVK13] presents a fully Bayesian model for CCA, shown in Figure 6.

Here, $\mathbf{V}^m$ constitutes the set of linear transformations for the m^{th} view. Again, the components of $\mathbf{v}_i$ are uncorrelated. [BJ06] presents an alternative interpretation of CCA where the latent vectors are themselves view-specific (shown in Figure 5) as well as proves Linear Discriminant Analysis(LDA) to be a special case of CCA. This model is extended to more than two views discussed at length in [VKKK14].

4.3.1 Inferring dimensionality of the latent space

Choosing k, the dimensionality of the latent space is a thorny problem. Too low a value of k would result in discarding some of the interesting correlation as noise, while too high a value of k captures spurious correlations. Automatic Relevance Determination (ARD), described in [Bis99], replaces the discrete selection of k with a continuous prior that discourages large factor loadings. While not inferring the actual value of k, the posterior of $\boldsymbol{\alpha}$, when concentrated at large values effectively switches off the extraneous dimensions, resulting in only relevant factors being al-

lowed to remain active [Bis99]. ARD is modelled as

$$p(\mathbf{V}|\boldsymbol{\alpha}) = \prod_k \mathcal{N}(\mathbf{V}_k|0, \alpha_k^{-1}\mathbf{I}),$$
$$p(\boldsymbol{\alpha}) = \prod_k \mathcal{G}(\alpha_k|a, b).$$

Each hyperparameter α_k controls the precision of k^{th} row of $\mathbf{V}$, such that as $\alpha_k \to \infty$, the entries of the k^{th} row would tend to zero, resulting in the factor being ignored. As far as we know, this little piece of ingenuity represents the first instance of automatic inference of k[5] in the word embeddings narrative.

While the extraneous rows are close enough to zero for practical purposes, sometimes we need a model that would make them *exactly* zero. One such alternative is use the spike-and-slab prior, where each entry in $\mathbf{V}$ is drawn from a two-component prior. One of these components, the *slab* corresponds to the delta distribution at zero while the other, the *spike* is drawn from the relatively uninformative Gaussian prior. The choice between either component is provided by the Bernoulli distribution. Formally, we define the model as [KVK13]

$$p(\mathbf{V}_k|\mathbf{H}_{mk}, \alpha_k) = \begin{cases} \mathcal{N}(0, \alpha_k^{-1}\mathbf{I}) & : \mathbf{H}_k = 1 \\ \delta_0 & : \mathbf{H}_k = 0 \end{cases},$$
$$p(\mathbf{H}_k) = \text{Bernoulli}(\pi),$$
$$p(\pi) = \text{Beta}(1, 1).$$

where δ_o is the Dirac delta function at 0. For classical CCA, the empirical approach to inferring the latent dimensionality is to construct a scree plot of canonical correlations against the dimension [GDMB08]. A sharp drop between two successive values indicates that rank corresponding to the higher of the two values is the latent dimensionality [GDMB08]. Another non-Bayesian alternative is to use k-fold cross validation techniques to estimate the value of k that best explains the validation data upon reconstruction [KVK13].

4.4 Shortcomings

The Bayesian models presented hitherto pose some trouble adapting to the data we consider, namely endless tuples of words. The words – represented as one-hot

[5]We hypothesise that the entropy of a language plays a major role in inferring the dimensionality of the embeddings; one can analytically reason about how chaotic a language is from the dimensionality inferred.

vectors – result in extremely sparse binary matrices.

1. [KVK13] and most other models assume Gaussian likelihood suitable only for continuous valued data. [Kla14] refined this by introducing the Polya-Gamma augmentation [PSW13] to approximate the logistic function, also easily extended to negative binomial distribution for over-dispersed count data.

2. The existing model treats each $\mathbf{x}_i$ as a unique entry. This is simply not true in the case of words where the entries are drawn from a vocabulary $\mathcal{V}$. Thus, the latent matrix, instead of being n-dimensional should have only as many entries as the vocabulary size.

3. As we shall see in the ensuing section, approximations of the logistic function result in inversion of $k \times k$ matrices. If we were to allocate a latent vector for each data point this would result in n inversions which is impractical to say the least. The model would not scale.

4. Finally, having latent vectors corresponding to each token requires us to come up with reasonable schemes to approximate the embedding corresponding to each unique word by considering all appearances of such a word in the data stream. This is hard owing to the transience of data streams.

Taking these above concerns into account, we detail the factorisation model of [PKW14] which replaces the Gaussian likelihood with the logistic approximation described in the next section.

4.5 Logistic Approximation

Consider the logistic function $\sigma(x) = \frac{1}{1+\exp(-x)}$. As a likelihood function, this proves inconvenient as it destroys the conjugacy[6] existent in the Gaussian likelihood. The Jordan-Jaakkola bound [JGJS99] provides a Gaussian variational lower bound to logistic. With a little reordering,

$$\ln \sigma(x) = \frac{x}{2} - \ln\left(\exp\left(\frac{x}{2}\right) + \exp\left(-\frac{x}{2})\right)\right).$$

[6]It is entirely possible to replace the conjugate updates that result from the approximation with non-conjugate updates where the logistic likelihood is preserved. Although the asymptotic normality of the posterior ensures that posterior updates can be approximated by a multivariate normal with centred at the mode with a inverse Fisher matrix covariance, the computational cost arising as out of numerical quadratures and the complexity of the resulting derivatives serve as sufficient disincentives.

Let $f(x) = \ln(\exp(\frac{x}{2}) + \exp(-\frac{x}{2}))$. Since $f(x)$ is a concave function in x^2 (the second derivative being non-negative), a tangent to the $f(x)$ – approximated by the first order Taylor expansion in x^2 – is a global lower bound as shown in Figure 8. That is, for $x \in \Re$,

$$f(x) \geq f(\xi) + \frac{\partial f(\xi)}{\partial \xi^2}(x^2 - \xi^2)$$

where ξ is the variational parameter to optimise. The bound is exact whenever

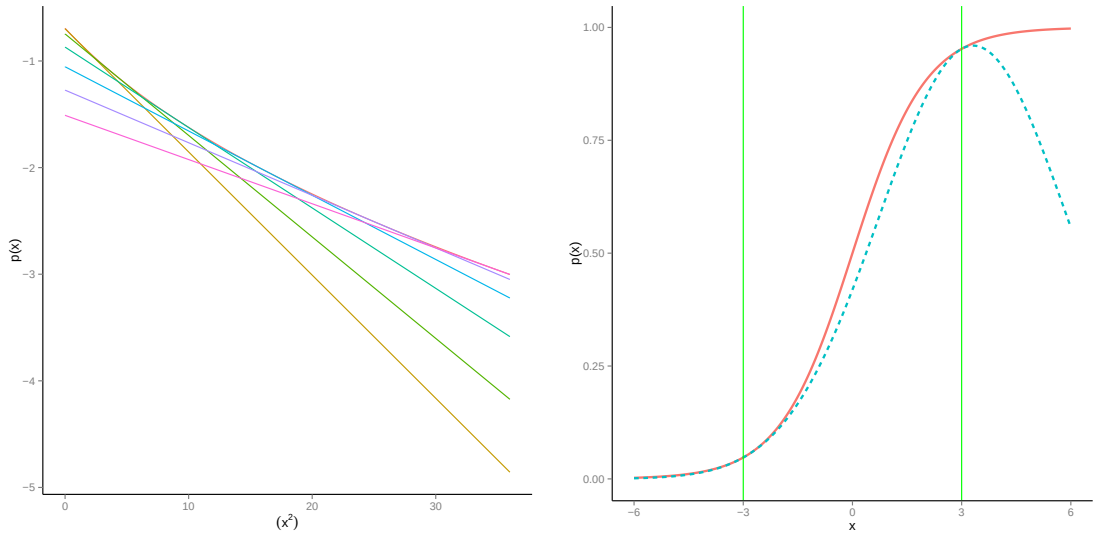


Figure 7: Left: $f(x)$, a convex function in x^2 is lower bounded by its first order Taylor approximation, shown here for different values of ξ^2. Right: The logistic sigmoid function $\sigma(x)$ (shown in red) is lower bounded by the Gaussian (shown in blue). The approximation is exact when $\xi = \pm 3$ (shown in green).

$\xi^2 = x^2$. We state that $\lambda(\xi) = \frac{\partial f(\xi)}{\partial \xi^2} = \frac{1}{4\xi}\tanh(\frac{\xi}{2})$ and,

$$f(x) \geq \sigma(\xi) - \frac{x}{2} + \lambda(\xi)(x^2 - \xi^2).$$

Thus, the lower bound for $\sigma(x)$ is given as

$$\sigma(x) \geq \exp\left(\frac{\xi + x}{2} + \lambda(\xi)(x^2 - \xi^2)\right).$$

One could as well trade off some of the accuracy and tightness of the Jaakkola bound for efficiency by using the Bohning bound [Boh92], a quadratic alternative. The Bohning bound forms a second order Taylor approximation to the $\ln \sigma(x)$ function at ξ, and replaces the Hessian $\mathbf{H}(\xi)$ with a constant matrix $\mathbf{A}$ such that $\mathbf{A} - \mathbf{H}(\xi)$ is positive definite for all ξ. While the Hessian approximation makes the Bohning

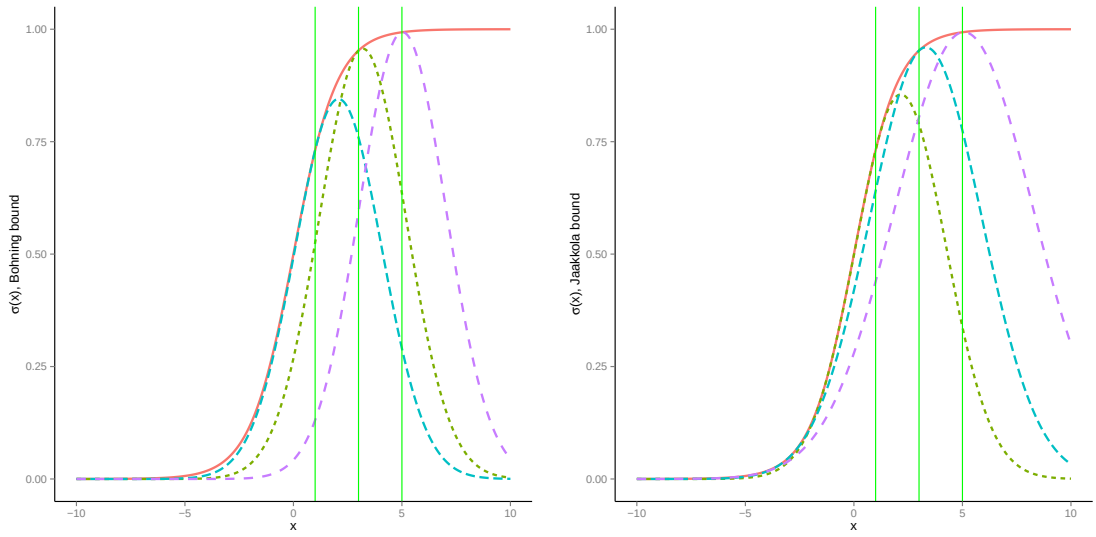


Figure 8: Left: The Bohning approximation to $\sigma(x)$ at $\xi = (1, 3, 5)$ the curvature of which is determined by $\mathbf{A}$. The Jaakkola approximation shown at $\xi = \pm(1, 3, 5)$ provides a tighter bound, but is computationally expensive. We use the Jaakkola bound in our model.

bound computationally efficient, it results in fixed curvatures unlike the Jaakkola bound. For this thesis, we will stick to the tried-and-tested Jaakkola bound instead of the Bohning bound as we are unsure of the practical implications it entails.

4.6 The Case For Logistic Assumption

A word w and its contexts $c \in C(w)$ can be modelled as entries in a sparse vector of dimension $1 \times |\mathcal{V}|$ where $|\mathcal{V}|$ is the size of the vocabulary. The vector $\mathbf{x}_w$ for w is set to one for all $c \in C(w)$ and to zero otherwise. This is traditionally know in the word embeddings literature as the *one-hot* representation and has been consistently employed and borne out by good results [TRB10]. Arguably, this may not be the best way to represent contextual information. For instance, probabilistic data structures such as the bloom filters [MU05] could be used to concisely express the same information with reasonable accuracy. However, it is usually the case that the sparse vectors for a given w are represented as a stream (w, c) pairs for $c \in C(W)$ with $c \notin C(w)$ implicitly assumed to be zero.

The usual assumption of a Gaussian generated data with unique latent entries for each observation, $\mathbf{x}_w \sim \mathcal{N}(\mathbf{u}_w^\top \mathbf{V}, \tau\mathbf{I})$ results in a bloated sparse matrix of size $|\mathcal{D}| \times |\mathcal{V}|$ where $|\mathcal{D}|$ is the number of distinct tokens in the data set $\mathcal{D}$. It is easy to see the

Gaussian assumption fails to hold when $\mathbf{x}_w$ is a sparse binary vector. Furthermore, besides making the running time dependent on the vocabulary size $|\mathcal{V}|$, this model also confuses the non-occurrence of the context $c' \notin C(w)$ in an entry with its express absence and models it as a negative instance (w, c'). This is not necessarily true, as a context c not found in a particular occurrence of $w \in \mathcal{D}$ may happen to appear in a different occurrence of w. The drawbacks of this may not be as pronounced were we to model each observed entry $\mathbf{x}_w$ in $\mathcal{D}$ with a corresponding latent entry $\mathbf{u}_w$.

However, we recognise that tokens in $\mathcal{D}$ are not unique, but are instead all drawn from $\mathcal{V}$. That is, each observed token $w \in \mathcal{D}$ corresponds to a unique word in $\mathcal{V}$. Therefore, it seems more fitting to model all occurrences of $w \in \mathcal{D}$ as a unique latent entry $\mathbf{u}_w$ instead of $|w \in \mathcal{D}|$ distinct latent entries. In such cases, the Gaussian model performs abysmally because we begin to unnecessarily (and, indeed, quite wrongly) penalise the unique $\mathbf{u}_w$ for non-occurrences of the contexts for a given word token.

If we were to model each entry j in $\mathbf{x}_i$ by the logistic function as,

$$x_{ij} = \begin{cases} \sigma(\mathbf{u}_i^\top \mathbf{v}_j) & : x_{ij} = 1 \\ 1 - \sigma(\mathbf{u}_i^\top \mathbf{v}_j) & : x_{ij} = 0 \end{cases}$$

it would alleviate several problems. Firstly, this assumption is a more faithful representation[7] of the data generation process. Secondly, it offers us a great deal of flexibility in deciding which absences of contexts we chose to cast as negative instances and which we do not. Thirdly, it is computationally efficient and is dependent only on the window size W plus the number of negative instances as opposed to the size of the entire vocabulary $\mathcal{V}$ in the previous model. Finally, because the logistic assumption preempts conjugacy we use the normal approximation discussed in the previous section to replace it.

This concludes our defence of the use of the logistic link function for factorising sparse binary matrices. In the following sections we discuss at length a Bayesian model that overcomes the several barriers enumerated so far.

4.7 The Generative Story

This and the following sections draw some ideas from [PKW14]. To remain true to our assumption of truly unsupervised learning, let us assume we are presented with a corpus of natural language text, where sentences are tokenised by $([\backslash .?!]^+\backslash s^+)$

[7] *All models are wrong, but some are useful.* George E.P. Box

and words by ([[:punct:]]*\s+). This constitutes all the preprocessing we shall do[8]. Let the input arrive as tuples of the form $(\mathbf{l}, w, \mathbf{r})$, where $\mathbf{l}$ and $\mathbf{r}$ represent the left and the right contexts of the word w. We will generically represent a single instance of the context as c. Let $\mathbf{U}, \mathbf{V} \in \Re^{k \times d}$. We construct a hash table to map w to $\mathbf{u}_w$ and c to $\mathbf{v}_c$.

Suppose that we pick w with probability π_w and c with probability ψ_c. Note that we could better model the probability of context ψ_c by making it conditional on w, but we refrain from doing so. The probability of observing this tuple is $\sigma(\mathbf{u}_w^\top \mathbf{v}_c)$ and of censoring it is $1-\sigma(\mathbf{u}_w^\top \mathbf{v}_c)$. Let us also introduce normal biases b_w and b_c. Intuitively, if w occurs in a wide variety of contexts, such as *the* or *a*, it is modelled by a large positive bias b_w, while a negative bias indicates w occurs in very specific contexts. Since we do not use it in our experiments, we shall limit ourselves to describing the idea. Let $\mathcal{D} = (T, w, c)$ constitute the observed set of tuples and $\mathcal{D}' = F$, the set of *unobserved*, censored negative samples. We do not as yet know the value of $\mathcal{D}'$, but according to [MCCD13] for large corpora, setting the ratio $r = \frac{\mathcal{D}}{\mathcal{D}'}$ to 5 appeared to work well. Let $\boldsymbol{\theta} = (\mathbf{U}, \mathbf{V}, \mathbf{b}, \boldsymbol{\pi}, \boldsymbol{\psi})$ The probability of coming across a tuple $(w, c) \in \mathcal{D}$ is

$$p(T, w, c|\boldsymbol{\theta}) = \pi_w \psi_c \sigma(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c)$$

while the probability of censoring an unknown tuple $(w', c') \in \mathcal{D}'$ becomes a mixture over d^2 components,

$$\begin{aligned} p(F|\boldsymbol{\theta}) &= \int p(o = F|w', c', \boldsymbol{\theta}) p(w'|\boldsymbol{\pi}) p(c'|\boldsymbol{\psi})\, dw'\, dc' \\ &= \int (1 - \sigma(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c) p(w'|\boldsymbol{\pi}) p(c'|\boldsymbol{\psi})\, dw'\, dc'. \end{aligned}$$

We model $\mathbf{U}$ as $\mathcal{N}(0, \mathbf{I})$, and $\mathbf{V}$ as a zero mean Gaussian with ARD priors. The censored data stream forms a negative background that discourages large values for $\sigma(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c)$. The joint likelihood of $D = (\mathcal{D} \cup \mathcal{D}')$ is therefore given by

$$p(\mathcal{D}, \boldsymbol{\theta}) = \left(\prod_{(w,c) \in \mathcal{D}} \sigma(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c) \right) \left(\prod_{(w,c) \in \mathcal{D}'} (1 - \sigma(\mathbf{u}_w^\top \mathbf{v}_c) + b_w + b_c) \right).$$

We shall use $\boldsymbol{\Theta} = ((w, c) \in \mathcal{D}', \boldsymbol{\theta})$ to denote the entire parameter set. Having described the intuition behind the model, we present it in the following section.

[8]Of course, this might be a poor idea for agglutinative language families like Finno-Ugric and Dravidian which require further rule-based but exhaustible preprocessing.

4.8 The Model

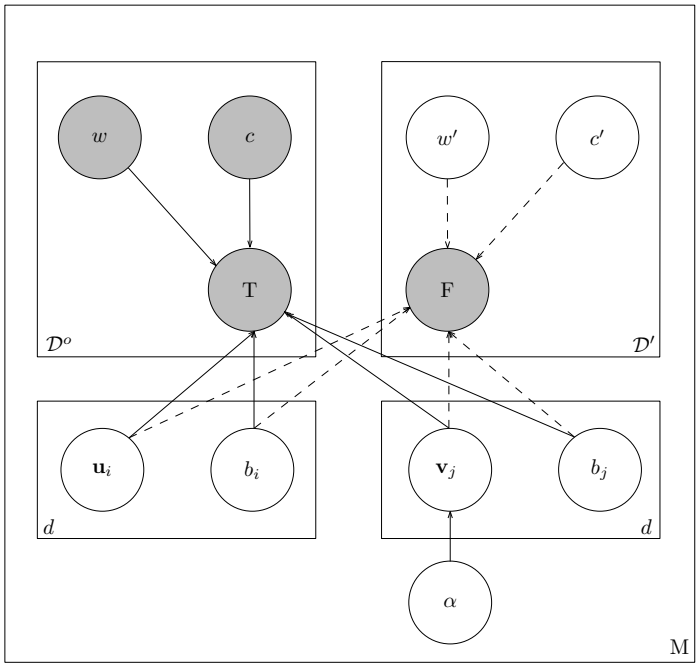


Figure 9: Plate diagram of the model: We draw (w, c) pairs from a uniform distribution. We observe $\mathcal{D}$ such pairs with probability $\sigma(\mathbf{u}_w^\top \mathbf{v}_c)$ while censoring D' such pairs with probability $(1 - \sigma(\mathbf{u}_w^\top \mathbf{v}_c))$ assuming we know what the censored pairs were. We employ ARD priors for dimensionality selection.

The model and its priors are shown in Figure 9. The joint probability is given by,

$$p(\mathcal{D}, \mathbf{\Theta}) = \prod_{(w,c)\in\mathcal{D}} \sigma(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c) \prod_{(w,c)\in\mathcal{D}'} (1 - \sigma(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c) \times p(\mathbf{U}) \cdot p(\mathbf{V}) \cdot p(\mathbf{b}) \cdot p(\boldsymbol{\alpha})$$

where,

$$\begin{aligned} p(\mathbf{U}) &= \prod_w \mathcal{N}(\mathbf{u}_w|0, \mathbf{I}), \\ p(\mathbf{V}|\boldsymbol{\alpha}) &= \prod_{m,k} \mathcal{N}(\mathbf{v}_k|0, \frac{1}{\boldsymbol{\alpha_k}}\mathbf{I}), \\ p(\boldsymbol{\alpha}) &= \prod_{m,k} \mathcal{G}(\alpha_k|a, b), \\ p(\mathbf{b}_w) &= \prod_w \mathcal{N}(b_w|0, \tau_w), \\ p(\mathbf{b}_c) &= \prod_c \mathcal{N}(b_c|0, \tau_c). \end{aligned}$$

4.9 Variational Approximation

Let $(w, c) \in \mathcal{D}$ be the set of observed data. The complete log likelihood of the model in Figure 9 is

$$\begin{aligned}
\mathcal{L}(q) &\propto \ln p(\mathcal{D}|\boldsymbol{\Theta}) + \ln p(\boldsymbol{\Theta}) \\
&= \sum_{(w,c)\mathcal{D}} \mathbb{E}_q \Bigg[\ln \sigma(\xi_{wc}) - \lambda(\xi_{wc}) \left((\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c)^2 - \xi_{wc}^2 \right) \\
&\qquad + \frac{1}{2}(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c) - \frac{\xi_{wc}}{2} \Bigg] \\
&+ \sum_{(w,c)\in\mathcal{D}'} \mathbb{E}_q[w_{d'} c_{d'}] \mathbb{E}_q \Bigg[\ln \sigma(\xi_{wc}) - \lambda(\xi_{wc}) \left((\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c)^2 - \xi_{wc}^2 \right) \\
&\qquad - \frac{1}{2}(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c) - \frac{\xi_{wc}}{2} \Bigg] \\
&+ \sum_w \mathbb{E}_q[\ln p(\mathbf{u}_w)] + \sum_k \mathbb{E}_q[\ln p(\mathbf{v}_k)] + \sum_k \mathbb{E}_q[\ln p(\alpha_k)].
\end{aligned}$$

Approximating the intractable $\ln p(\boldsymbol{\Theta}|\mathcal{D})$ with $q(\boldsymbol{\Theta})$, we note that the lower bound to optimise becomes,

$$\ln p(\mathcal{D}) \geq \ln \int p_\xi(\mathcal{D}, \boldsymbol{\Theta}) d\,\boldsymbol{\Theta} \geq \ln \int q(\boldsymbol{\Theta}) \frac{p_\xi(\mathcal{D}, \boldsymbol{\Theta})}{q(\boldsymbol{\Theta})} d\,\boldsymbol{\Theta} = \mathcal{L}(q),$$

where $p_\xi(\mathcal{D}, \boldsymbol{\Theta})$ is joint distribution when the Gaussian lower bound replaces the logistic function so as to preserve conjugacy. We assume the following mean-field factorisation to approximate the posterior as

$$q(\boldsymbol{\Theta}) = \prod_w q(\mathbf{u}_w) q(b_w) \prod_c q(\mathbf{v}_c|\boldsymbol{\alpha}) q(b_c).$$

Finally the updates for unconstrained optimisation over each of the factor distributions are

$$q(\boldsymbol{\Theta}_i) \propto \exp\left(\mathbb{E}_{q_{-i}} [\ln p(D, \boldsymbol{\Theta})] \right) + \text{const}$$

or, equivalently

$$\ln q(\boldsymbol{\Theta}_i) \propto \mathbb{E}_{q_{-i}} [\ln p(D, \boldsymbol{\Theta})] + \text{const}.$$

For the rest of this section we shall derive the variational updates for each of the parameters. For the sake of clarity, we shall state the updates as is leaving further optimisation to *Section 5*. Equations of relevance are starred.

4.9.1 Gaussian Updates

Consider $q(\mathbf{U})$ of the form $\prod_w q(\mathbf{u}_w) = \prod_w \mathcal{N}(\mathbf{u}_w|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$. The contribution of $\mathbf{u}_w$ towards $\mathcal{L}$ is given as

$$\begin{aligned}\ln q(\mathbf{u}_w) \propto & \sum_{(w,c)\in\mathcal{D}} \mathbb{E}_q\left[\lambda(\xi_{wc})\left((\mathbf{u}_w{}^\top\mathbf{v}_c + b_w + b_c)^2\right) + \frac{1}{2}(\mathbf{u}_w^\top\mathbf{v}_c + b_w + b_c)\right] \\ & + \sum_{(w,c)\in\mathcal{D}'} \mathbb{E}_q\left[-\lambda(\xi_{wc})\left((\mathbf{u}_w^\top\mathbf{v}_c + b_w + b_c)^2\right) - \frac{1}{2}(\mathbf{u}_w^\top\mathbf{v}_c + b_w + b_c)\right] \\ & - \frac{1}{2}\mathrm{tr}\left(\mathbb{E}_q[\mathbf{u}_w\mathbf{u}_w^\top]\mathbf{I}\right).\end{aligned}$$

Rearranging,

$$\begin{aligned}\ln q(\mathbf{u}_w) \propto & -\frac{1}{2}\mathrm{tr}\,\mathbb{E}_q[\mathbf{u}_w\mathbf{u}_w^\top]\left(\mathbf{I} + \sum_{(w,c)\in\mathcal{D}} 2\lambda(\xi_{wc})\mathbb{E}_q[\mathbf{v}_c\mathbf{v}_c{}^\top] + \sum_{(w,c)\in\mathcal{D}'} 2\lambda(\xi_{wc})\mathbb{E}_q[\mathbf{v}_c\mathbf{v}_c{}^\top]\right) \\ & + \mathbb{E}_q[\mathbf{u}_w^\top]\left\{\sum_{(w,c)\in\mathcal{D}}\left(\frac{1}{2} + 2\lambda(\xi_{wc})\mathbb{E}_q(b_w + b_c)\right)\right. \\ & \left. - \sum_{(w,c)\in\mathcal{D}'}\left(\frac{1}{2} + 2\lambda(\xi_{wc})\mathbb{E}_q(b_w + b_c)\right)\mathbb{E}_q[\mathbf{v}_c]\right\}.\end{aligned}$$

This clearly has a the form of a Gaussian posterior and the updates for $q(\mathbf{u}_w)$ are given by

$$\boldsymbol{\Sigma}_w = \left(\mathbf{I} + \sum_{(w,c)\in\mathcal{D}} 2\lambda(\xi_{wc})\mathbb{E}_q[\mathbf{v}_c\mathbf{v}_c{}^\top] + \sum_{(w,c)\in\mathcal{D}'} 2\lambda(\xi_{wc})\mathbb{E}_q[\mathbf{v}_c\mathbf{v}_c{}^\top]\right)^{-1}, \qquad (\star)$$

$$\begin{aligned}\boldsymbol{\mu}_w = \boldsymbol{\Sigma}_w & \left\{\sum_{(w,c)\in\mathcal{D}}\left(\frac{1}{2} + 2\lambda(\xi_{wc})\mathbb{E}_q(b_w + b_c)\right)\right. \\ & \left. - \sum_{(w,c)\in\mathcal{D}'}\left(\frac{1}{2} + 2\lambda(\xi_{wc})\mathbb{E}_q(b_w + b_c)\right)\mathbb{E}_q[\mathbf{v}_c]\right\}. \qquad (\star)\end{aligned}$$

This completes the updates for $q(\mathbf{u_w})$. The update of the parameters for each $\mathbf{u}_w$ is an embarrassingly parallel task – that is, the each $\mathbf{u}_w$ can be computed independently of others with no communication – and multithreading speeds up the inference greatly.

The biases again have the Gaussian form $q(b_i) = \prod_i \mathcal{N}(b_i, \mu_i, \tau_i)$ where i is either w or c. The contribution of b_w toward $\mathcal{L}(q)$ is given as

$$\begin{aligned}
\ln q(b_w) \propto & \sum_{(w,c)\in\mathcal{D}} \mathbb{E}_q \left[\lambda(\xi_{wc}) \left(({\mathbf{u}_w}^\top \mathbf{v}_c + b_w + b_c)^2 \right) + \frac{1}{2}(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c) \right] \\
& + \sum_{(w,c)\in\mathcal{D}'} \mathbb{E}_q \left[-\lambda(\xi_{wc}) \left((\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c)^2 \right) - \frac{1}{2}(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c) \right] \\
& - \frac{1}{2} \left(\mathbb{E}_q[b_w^2] \tau_w \right) \\
= & -\frac{1}{2}\mathbb{E}_q[b_w^2] \left(\tau_b + \sum_{(w,c)\in\mathcal{D}} 2\lambda(\xi_{wc}) + \sum_{(w,c)\in\mathcal{D}'} 2\lambda(\xi_{wc}) \right) \\
& + \mathbb{E}_q[b_w] \left\{ \sum_{(w,c)\in\mathcal{D}} \left(\frac{1}{2} + 2\lambda(\xi_{wc}) \mathbb{E}_q(\mathbf{u}_w^\top \mathbf{v}_c + b_c) \right) \right. \\
& \qquad \left. - \sum_{(w,c)\in\mathcal{D}'} \left(\frac{1}{2} + 2\lambda(\xi_{wc}) \mathbb{E}_q(\mathbf{u}_w^\top \mathbf{v}_c + b_c) \right) \right\}.
\end{aligned}$$

We now see the updates for $q(b_w)$ are given by

$$\tau_w = \left(\tau_b + \sum_{(w,c)\in\mathcal{D}} 2\lambda(\xi_{wc}) + \sum_{(w,c)\in\mathcal{D}'} 2\lambda(\xi_{wc}) \right), \qquad (\star)$$

$$\begin{aligned}
\mu_w = \tau_w^{-1} & \left\{ \sum_{(w,c)\in\mathcal{D}} \left(\frac{1}{2} + 2\lambda(\xi_{wc}) \mathbb{E}_q(\mathbf{u}_w^\top \mathbf{v}_c + b_c) \right) \right. \\
& \left. - \sum_{(w,c)\in\mathcal{D}'} \left(\frac{1}{2} + 2\lambda(\xi_{wc}) \mathbb{E}_q(\mathbf{u}_w^\top \mathbf{v}_c + b_c) \right) \right\}. \qquad (\star)
\end{aligned}$$

Similarly, the updates for $q(b_c)$ are given by

$$\tau_c = \left(\tau_c + \sum_{(w,c)\in\mathcal{D}} 2\lambda(\xi_{wc}) + \sum_{(w,c)\in\mathcal{D}'} 2\lambda(\xi_{wc}) \right), \qquad (\star)$$

$$\begin{aligned}
\mu_c = \tau_c^{-1} & \left\{ \sum_{(w,c)\in\mathcal{D}} \left(\frac{1}{2} + 2\lambda(\xi_{wc}) \mathbb{E}_q(\mathbf{u}_w^\top \mathbf{v}_c + b_w) \right) \right. \\
& \left. + \sum_{(w,c)\in\mathcal{D}'} \left(\frac{1}{2} + 2\lambda(\xi_{wc}) \mathbb{E}_q(\mathbf{u}_w^\top \mathbf{v}_c + b_w) \right) \right\}. \qquad (\star)
\end{aligned}$$

4.9.2 ARD Updates

The ARD updates look very similar to those of the Gaussian updates except that the prior for $\mathbf{v_c}$ takes the form

$$\mathbb{E}_q[\ln p(\mathbf{v}_c)] = -\frac{1}{2}\text{tr}\ \left(\mathbb{E}_q[\mathbf{v}_c\mathbf{v}_c{}^\top]\text{diag}(\boldsymbol{\alpha})\right).$$

The contribution of $\mathbf{v}_c$ towards $\mathcal{L}$ is given by

$$\begin{aligned}\ln q(\mathbf{v}_c) \propto &\sum_{(w,c)\in\mathcal{D}} \mathbb{E}_q\left[\lambda(\xi_{wc})\left((\mathbf{u}_w{}^\top\mathbf{v}_c + b_w + b_c)^2\right) + \frac{1}{2}(\mathbf{u}_w^\top\mathbf{v}_c + b_w + b_c)\right] \\ &+ \sum_{(w,c)\in\mathcal{D}'} \mathbb{E}_q\left[-\lambda(\xi_{wc})\left((\mathbf{u}_w^\top\mathbf{v}_c + b_w + b_c)^2\right) - \frac{1}{2}(\mathbf{u}_w^\top\mathbf{v}_c + b_w + b_c)\right] \\ &- \frac{1}{2}\text{tr}\ \left(\mathbb{E}_q[\mathbf{v}_c\mathbf{v}_c{}^\top]\text{diag}(\boldsymbol{\alpha})\right).\end{aligned}$$

Rearranging,

$$\begin{aligned}\ln q(\mathbf{v}_c) \propto &-\frac{1}{2}\text{tr}\,\mathbb{E}_q[\mathbf{v}_c\mathbf{v}_c^\top]\left(\text{diag}(\boldsymbol{\alpha}) + \sum_{(w,c)\in\mathcal{D}} 2\lambda(\xi_{wc})\mathbb{E}_q[\mathbf{u}_w\mathbf{u}_w{}^\top] + \sum_{(w,c)\in\mathcal{D}'} 2\lambda(\xi_{wc})\mathbb{E}_q[\mathbf{u}_w\mathbf{u}_w{}^\top]\right) \\ &+ \mathbb{E}_q[\mathbf{v}_c^\top]\left\{\sum_{(w,c)\in\mathcal{D}}\left(\frac{1}{2} + 2\lambda(\xi_{wc})\mathbb{E}_q(b_w + b_c)\right)\right. \\ &\left. - \sum_{(w,c)\in\mathcal{D}'}\left(\frac{1}{2} + 2\lambda(\xi_{wc})\mathbb{E}_q(b_w + b_c)\right)\mathbb{E}_q[\mathbf{u}_w]\right\}.\end{aligned}$$

The updates for $q(\mathbf{v}_c)$ are then given by

$$\boldsymbol{\Sigma}_c = \left(\text{diag}(\boldsymbol{\alpha}) + \sum_{(w,c)\in\mathcal{D}} 2\lambda(\xi_{wc})\mathbb{E}_q[\mathbf{u}_w\mathbf{u}_w{}^\top] + \sum_{(w,c)\in\mathcal{D}'} 2\lambda(\xi_{wc})\mathbb{E}_q[\mathbf{u}_w\mathbf{u}_w{}^\top]\right)^{-1}, \quad (\star)$$

$$\begin{aligned}\boldsymbol{\mu}_c = \boldsymbol{\Sigma}_c &\left\{\sum_{(w,c)\in\mathcal{D}}\left(\frac{1}{2} + 2\lambda(\xi_{wc})\mathbb{E}_q(b_w + b_c)\right)\right. \\ &\left. - \sum_{(w,c)\in\mathcal{D}'}\left(\frac{1}{2} + 2\lambda(\xi_{wc})\mathbb{E}_q(b_w + b_c)\right)\mathbb{E}_q[\mathbf{u}_w]\right\}. \quad (\star)\end{aligned}$$

This completes the updates for $\mathbf{v_c}$. Again, we see that the updates are embarrassingly parallel.

4.9.3 Gamma Updates

The contribution of $\boldsymbol{\alpha}_k$ to $\mathcal{L}(q)$ is given by

$$\begin{aligned}\mathcal{L}_\xi(q) &= \mathbb{E}_q\left[\frac{d}{2}\sum_k \ln \alpha_m^k - \frac{1}{2}\sum_k \alpha_k \mathbf{v}_k{}^\top \mathbf{v}_k + \sum_k \left((a-1)\ln\alpha_k - b\alpha_k\right)\right] \\ &= \mathbb{E}_q\left[\left(\frac{d}{2} + (a-1)\right)\sum_k \ln\alpha_k - \left(\frac{1}{2}\mathbf{v}_k{}^\top \mathbf{v}_k + b\right)\sum_k \alpha_k\right].\end{aligned}$$

The updates are of $q(\boldsymbol{\alpha})$ are

$$a' = a + \frac{d}{2}, \tag{$\star$}$$

$$b' = b + \frac{1}{2}\sum_k \mathbb{E}_q\left[\mathbf{v}_k{}^\top \mathbf{v}_k\right]. \tag{$\star$}$$

The value for $\sum_k \mathbb{E}_q\left[\mathbf{v}_k{}^\top \mathbf{v}_k\right]$ is given by $\sum_k \operatorname{tr}\, \mathbb{E}_q\left[\mathbf{v}_k \mathbf{v}_k{}^\top\right]$.

4.9.4 Logistic Bounds

[PKW14] treats the logistic bound separately for the observed values (i, j) while tying down the bound for censored observations to ξ^*. Consider a given view m. By differentiating its contribution to $\mathcal{L}(q)$ w.r.t ξ_{ij}, the optimal bound becomes

$$\mathcal{L}_\xi(q) = \mathbb{E}_q\left[\ln\sigma(\xi_{ij}) - \lambda(\xi_{ij})\left((\mathbf{u}_i^\top \mathbf{v}_j + b_i + b_j)^2 - \xi_{ij}^2\right) + \frac{1}{2}(\mathbf{u}_i^\top \mathbf{v}_j + b_i + b_j) - \frac{\xi_{ij}}{2}\right].$$

Differentiating w.r.t ξ_{ij} and maximising,

$$\begin{aligned}\frac{\partial \mathcal{L}_\xi(q)}{\partial \xi_{wc}} &= \lambda'(\xi_{wc})\frac{\partial}{\partial \lambda(\xi_{wc})}\mathbb{E}_q\Big[\ln\sigma(\xi_{wc}) - \lambda(\xi_{wc}) \\ &\qquad \left((\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c)^2 - \xi_{wc}^2\right) + \frac{1}{2}(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c) - \frac{\xi_{wc}}{2}\Big] = 0 \\ \implies \qquad 0 &= \lambda'(\xi_{wc})(\mathbb{E}_q[(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c)^2] - \xi_{wc}^2) \\ \implies \qquad \xi_{wc}^2 &= \mathbb{E}_q[(\mathbf{u}_w^\top \mathbf{v}_c + b_w + b_c)^2]. \qquad (\star)\end{aligned}$$

Storing the logistic parameter for each unique (w, c) is prohibitively costly. Instead, we calculate it on the fly.

4.9.5 Biases

From the above equations it is not hard to see that dispensing with the biases leads to a reduction in the number of calculations required for an update. For example,

in the logistic updates, when the biases are done away with, the equation for ξ_{wc} simply reduces to

$$\xi_{wc} = \text{tr}(\mathbb{E}_q[\mathbf{u}_w \mathbf{u}_w^\top]\mathbb{E}_q[\mathbf{v}_c \mathbf{v}_c]).$$

We contend that since the biases possess just informative value by giving an estimate of the contextual nature of the word, they do not prove strictly necessary for our model, losing them does not affect the quality of the word embeddings. Thus, in our actual algorithms presented in Chapter 5 we simply choose to ignore them.

4.10 Negative Sampling

The idea of negative sampling [MCCD13] is to generate a so-called negative data stream $(w, c) \in \mathcal{D}'$ that helps push the value $\sigma(\mathbf{u}_w^\top \mathbf{v}_c)$ down [GL14]. In the absence of a negative stream, the likelihood for the observed data,

$$\arg\max_{\mathbf{v}_c\, \mathbf{v}_w} \sum_{w \in \mathcal{C},\ c \in C(w)} \sigma(\mathbf{v}_c^\top\, \mathbf{v}_w)$$

is maximised by trivially setting $\mathbf{u}_w = \mathbf{v}_c$ such that $\mathbf{u}_w^\top \mathbf{v}_c = k$ for a large enough k. This results in all the vectors looking the same. [GH12] first showed that by introducing artificially generated noise distinct from the observed data leads to consistent estimation of parameters in logistic regression. [MCCD13] borrowed this paradigm and showed that by pushing the vectors *away* from certain artificially generated combinations, one obtains non-trivial solutions. Thus the modified likelihood is

$$\arg\max_{\mathbf{v}_c\, \mathbf{v}_w} \left[\sum_{w \in \mathcal{C},\ c \in C(w)} \sigma(\mathbf{v}_c^\top\, \mathbf{v}_w) + \sum_{w \in \mathcal{C},\ c' \notin C(w)} \sigma(-\mathbf{v}_{c'}^\top\, \mathbf{v}_w) \right]$$

where $c' \notin C(w)$ is appropriately sampled from suitable distribution. This is the basic intuition behind negative sampling. Note that by fixing either $\mathbf{v}_c$ or $\mathbf{v}_w$, this model simply reduces to logistic regression.

We can employ a number of schemes for negative sampling the simplest being sampling from the vocabulary uniformly. While more complex schemes such as popularity based sampling as is common in recommendation system can be used, we found the uniform sampler to work just fine.

5 Scalable Bayesian Word Embeddings

> As a rule, I am very careful to be shallow and conventional where depth and originality are wasted.
>
> *Lucy Maud Montgomery*

Variational inference through alternating updates offers a quick and satisfactory solution although it does have a tendency to get stuck in local minima. That said, variational inference fails to scale up to large data sets – sweeping through the entire data set for each update of the parameters becomes infeasible in the long run. Recently [HBWP13] introduced a stochastic variant of variational inference, a scalable method which borrows techniques from stochastic optimisation [RM51] to optimise the objective function by obtaining noisy estimates of the gradient This overcomes the previously mentioned hurdle and helps handle massive data sets. In this chapter we offer two solutions to make our algorithm a practical one.

- In keeping with our promise of a stochastic and scalable algorithm we offer a purely SVI-type algorithm to scale up to any length of data.
- The second solution is a hybrid of batch and stochastic versions that uses a potpourri of multithreading, mini-batches and noisy downsampling to estimate a better approximation of the likelihood than one estimated by a purely SVI-based algorithm.

Before defining our algorithms, we briefly review some preliminaries for stochastic inference.

5.1 Natural Gradients

In this section we briefly review the usefulness of natural gradients for quick convergence. According to [HTRK08], in stochastic variational updates, the parameter space is not Euclidean. Thus the traditional gradient, that points in the direction of steepest gradient in the Euclidean case with an orthonormal coordinate system, is not accurate when the parameter space is a curved manifold with no orthonormal coordinates. This curved manifold is known as the Riemannian space. Let us suppose we aim at optimising an objective function with respect to a parametrised

probability distribution $q(\beta|\lambda)$. In case of a Euclidean space, the gradient $\nabla_\lambda f(\lambda)$ points in the same direction as

$$\arg\max f(\lambda + d\lambda) \qquad \|d\lambda\|^2 < \epsilon^2.$$

[HBWP13] suggests the use of summarised KL divergence as the natural measure of dissimilarity between two distributions

$$D_{KL}^{sym}(\lambda, \lambda') = \mathbb{E}_\lambda \left[\log \frac{q(\beta|\lambda)}{q(\beta|\lambda')}\right] + \mathbb{E}_\lambda \left[\log \frac{q(\beta|\lambda')}{q(\beta|\lambda)}\right].$$

Now let us replace the Euclidean metric with D_{KL}^{sym}, a Riemannien metric. Then, the direction of the steepest ascent becomes

$$\arg\max f(\lambda + d\lambda) \qquad D_{KL}^{sym}(\lambda, \lambda + d\lambda) < \epsilon.$$

Unlike the Euclidean gradient, the natural gradient according to [Ama98] points in the direction of steepest descent in the Riemannian space and the distance between the two distributions parametrised by λ and $\lambda + d\lambda$ is the change in D_{KL}^{sym}. For most of the probability distributions, this *warping* of the Euclidean manifold is done by multiplying the normal gradient with the inverse of the Fisher information matrix $\mathcal{V}$ where

$$\mathcal{V}(\lambda) = \mathbb{E}[\nabla_\lambda \ln q(\theta|\lambda)(\nabla_\lambda \ln q(\theta|\lambda))^\top].$$

In the exponential family of distributions, the Fisher information is found by taking the second the second derivative of the log normaliser A, $\frac{\partial^2 A(\lambda_i)}{\partial \lambda_i \partial \lambda_i^\top}$

Recalling from Section 2.3 the natural gradient for $\mathcal{L}(q)$ is given by

$$\tilde{\nabla}_{\lambda_l} \mathcal{L} = \mathcal{V}(\lambda_i)^{-1} \nabla_{\lambda_l} \mathcal{L}$$

where $\mathcal{V}(\lambda_i)^{-1}$ is the inverse of the Fisher information matrix. It is not hard to see that

$$\begin{aligned}
\tilde{\nabla}_{\lambda_l} \mathcal{L} &= \mathcal{V}(\lambda_i)^{-1} \nabla_{\lambda_l} \mathcal{L} \\
&= \mathcal{V}(\lambda_i)^{-1} (\mathcal{V}(\lambda_i) \mathbb{E}_q(\eta_a(x, \theta_{-i})) - \lambda_i \mathcal{V}(\lambda_i)) \\
&= \mathbb{E}_q[\eta_a(x, \theta_{-i})] - \lambda_i \\
&= 0 \\
\implies \lambda_i &= \sum_a \mathbb{E}_q(\eta_a(x, \theta_{-i}))
\end{aligned}$$

Moreover, if any factor in S_a is not in the conjugate exponential family, then

$$\begin{aligned}\tilde{\nabla}_{\lambda_l}\mathcal{L} &= \mathcal{V}(\lambda_i)^{-1}\nabla_{\lambda_l}\mathcal{L} \\ &= \mathcal{V}(\lambda_i)^{-1}(\nabla_{\lambda_l}\mathbb{E}_q[\ln p(\mathcal{D},\theta)] - \mathcal{V}(\lambda_i)\lambda_i) \\ &= \mathcal{V}(\lambda_i)^{-1}\nabla_{\lambda_l}\mathbb{E}_q[\ln p(\mathcal{D},\theta)] - \lambda_i) \\ &= 0 \\ \implies \lambda_i &= \mathcal{V}(\lambda_i)^{-1}\nabla_{\lambda_l}\mathbb{E}_q[\ln p(\mathcal{D},\theta)].\end{aligned}$$

We shall however restrict ourselves to the conjugate exponential family of distributions with the neat update

$$\lambda_i = \sum_a \mathbb{E}_q(\eta_a(x,\theta_{-i})).$$

5.2 Scaling up

Stochastic ascent optimises the objective function by obtaining noisy estimates of the gradient. Robbins-Munro[RM51] showed that if the step sizes ϵ_t at the t^{th} iteration satisfy the conditions

$$\sum_t \epsilon_t = \infty,$$
$$\sum_t \epsilon_t^2 < \infty$$

then λ^t will converge to either the global or local optimum with the following stochastic update

$$\lambda^{t+1} = \lambda^t + \epsilon_t\tilde{\nabla}_\lambda\mathcal{L}.$$

In other words if $\tilde{\nabla}_{\lambda_l}\mathcal{L}$ is a noisy estimate of the gradient and $\hat{\lambda}_i = \sum_a \mathbb{E}_q(\eta_a(x,\theta_{-i}))$ is the noisy natural parameter obtained, then

$$\lambda^{t+1} = (1-\epsilon_t)\lambda^t + \epsilon_t\hat{\lambda}_i.$$

While theoretically a single data point can be used to obtain a noisy estimate of the gradient by multiplying the gradient estimate of that single point by the total number of data points, in practice we tend to use gradient estimates from small batches of data times the number of batches to obtain a less noisy estimate of the gradient. This leads to a simple algorithm shown in Algorithm 2.

In most cases, the variational parameters can be conveniently split into local variables corresponding to each observation and global variables. Since any given local

Algorithm 2: Stochastic Variational Bayes.

input : The data set $\mathcal{D}$ and appropriately parametrised $q(\theta_i|\lambda_i)\ \forall i \in [1, w]$, the forget rate κ and the decay rate τ

output: $\lambda_i,\ \forall i$

while *true* **do**
  Sample a data point $d \in \mathcal{D}$ **for** $i \leftarrow 1$ **to** w **do**
    $\hat{\lambda}_i = |\mathcal{D}| \sum_a \mathbb{E}_q(\eta_a(d, \theta_{-i}))$
  $\lambda^{t+1} = (1 - \epsilon_t)\lambda^t + \epsilon_t \hat{\lambda}$
  $\epsilon_t = (1 + \tau)^{\kappa}$

variable is conditionally independent of all other local variables given the global variables, [HBWP13] proposes an algorithm that optimises the local parameters of a given batch to produce an intermediate global estimate. This estimate is weighted with the current estimates of the global parameters to produce a new global estimate.

As a case in point, [HlHG14] employs this clear-cut distinction to characterise a stochastic binary factorisation model $\mathbf{x}_i = \mathbf{u}_i^\top \mathbf{V} + \epsilon$ for each observation i. It is straightforward to see that $\mathbf{u}_i$ corresponds to the local parameter for each observation and the factor loading matrix $\mathbf{V}$ is global.

Our model, however, does not allow for local and global variational updates as expected in traditional stochastic inference; rather, we resort to semi-global parameter updates, with only those words updated that exist in the mini-batch under consideration. To elaborate, we choose only those rows of $\mathbf{U}$ and $\mathbf{V}$ that correspond to the unique vocabulary subset in the mini-batch to optimise and use the intermediate estimates to update our current global estimates corresponding to these rows alone. What this means is that we do not have the luxury of throwing away the local estimates of $\mathbf{u}_i$ after a global update of $\mathbf{V}$, but instead are forced to use these to update a semi-global $\mathbf{U}$ matrix.

5.2.1 Downsampling

Since the relationship between the words and their frequencies follows a power law, we find that a small number of words constitute a large percentage of the total tokens. Words, such as *the* and *a* or certain prepositions are inherently less informative than their rarer counterparts. For example, the pair *(cloudy, sky)* is of more value than *(the, sky)* as *the* occurs probably with every other word. [MCCD13] suggests

a policy of downsampling that aims to address the disparity between words by aggressively downsampling frequent words according to the following heuristic

$$p(w) = 1 - \sqrt{\frac{t}{f(w)}}$$

where t is the threshold, typically 10^{-5}, and $f(w)$ is the frequency of the words. This heuristic downsamples only those words whose frequency exceeds t, while leaving the rest untouched. Besides giving better quality embeddings, downsampling speeds up the algorithm significantly.

5.3 Algorithms

In this section, we present two novel algorithms that arise as a culmination of all our previous discussion. The updates in the algorithms roughly reflect those derived in Section 4.9.

5.3.1 Stochastic Version

We first provide the stochastic version of our model. For the sake of clarity we shall use $\mathbf{U}$ to denote all parameters associated with $\mathbf{U}$ such as its means $\mathbf{U}_\mu$, its precisions $\mathbf{U}_\Lambda$, its second moments $\mathbf{U}_{uu}$ and mean-times-precision $\mathbf{U}_\nu$.

Algorithm 3 takes as its input the current batch, the current estimates of the natural parameters of the two matrix factors and the forgetting and the delay rates, producing their converged values as the output. Line 1 calculates the factor for each word with which to multiply the current gradient to get a noisy estimate of the original gradient. Lines 3-11 show how $\mathbf{U}$ is updated with negative sampling. The 'real' parameters are recovered from the natural ones. The updates look similar for $\mathbf{V}$. Finally, the original natural parameters are gathered using Robbins-Munroe style updates. The algorithm thus meets our promise of a stochastic and a scalable variational technique.

5.3.2 Multithreaded Version

Now consider Algorithm 4 which in many ways seems similar to the stochastic version in Algorithm 3.

Algorithm 3: Stochastic Variational Inference for Word Embeddings

input : *batch*, $\mathbf{U}$, $\mathbf{V}$, κ, τ, n

output: $\mathbf{U}$ and $\mathbf{V}$

1 $\hat{\mathbf{U}} \leftarrow \mathbf{U}$

2 $\hat{\mathbf{V}} \leftarrow \mathbf{V}$

3 $\boldsymbol{\beta} \leftarrow \frac{\text{Total count of each item}}{\text{Count of each item in current batch}}$

4 **for** *iter in niter* **do**

5 $\quad \mathbf{U}_\nu, \mathbf{U}_\Lambda \leftarrow 0$

6 $\quad \mathbf{V}_{vv} \leftarrow \mathbf{V}_\mu \mathbf{V}_\mu^\top + \mathbf{V}_\Lambda^{-1}$

7 $\quad$ **foreach** *w in batch* **do**

8 $\quad\quad$ **foreach** *c in window(w)* **do**

9 $\quad\quad\quad \mathbf{U}_\nu[w] \leftarrow \mathbf{U}_\nu[w] + 0.5 \cdot \mathbf{V}_\mu[c]$

10 $\quad\quad\quad \mathbf{U}_\Lambda[w] \leftarrow \mathbf{U}_\Lambda[w] + 2 \cdot \lambda_{w,c} \cdot \mathbf{V}_{vv}[c]$

11 $\quad\quad$ **foreach** *c in sample(n)* **do**

12 $\quad\quad\quad \mathbf{U}_\nu[w] \leftarrow \mathbf{U}_\nu[w] - 0.5 \cdot \mathbf{V}_\mu[c]$

13 $\quad\quad\quad \mathbf{U}_\Lambda[w] \leftarrow \mathbf{U}_\Lambda[w] + 2 \cdot \lambda_{w,c} \cdot \mathbf{V}_{vv}[c]$

14 $\quad$ **foreach** *w in n* **do**

15 $\quad\quad \mathbf{U}_\nu[w] \leftarrow \boldsymbol{\beta}[w]\mathbf{U}_\nu[w]$

16 $\quad\quad \mathbf{U}_\Lambda[w] \leftarrow \boldsymbol{\beta}[w]\mathbf{U}_\Lambda[w] + \mathbf{I}$

17 $\quad\quad \mathbf{U}_\mu[w] \leftarrow \mathbf{U}_\Lambda[w]^{-1}\mathbf{U}_\nu[w]$

18 $\quad$ *Similar updates for* $\mathbf{V}$*. Except, instead of adding* $\mathbf{I}$ at **16** one adds $diag(\boldsymbol{\alpha})$*, the ARD priors.*

19 **for** *w in n* **do**

20 $\quad \mathbf{U}[w] \leftarrow (1 - \boldsymbol{\epsilon}[w])\hat{\mathbf{U}}[w] + \boldsymbol{\epsilon}[w]\mathbf{U}[w]$

21 $\quad \mathbf{V}[w] \leftarrow (1 - \boldsymbol{\epsilon}[w])\hat{\mathbf{V}}[w] + \boldsymbol{\epsilon}[w]\mathbf{V}[w]$

22 $\quad \boldsymbol{\epsilon}[w] \leftarrow (\boldsymbol{\beta}[w] + \kappa)^\tau$

It takes as its input, two Gaussian distributed matrices factorised by rows and outputs their converged values. The algorithm creates a number of threads each working in a different part of the file, to obtain the unscaled noisy estimate for the gradient. These are finally combined into the 'true' estimate of the gradient. Arguably, this estimate is far more accurate than the stochastic gradient estimate. We use mutex locks to prevent race conditions. While this is not a problem for infrequent words, words occurring very frequently do result in race conditions making

Algorithm 4: Multithreaded Version

input : *data*, $\mathbf{U}$, $\mathbf{V}$, number of threads, n
output: $\mathbf{U}$ and $\mathbf{V}$

1 **for** *iter in niter* **do**
2 $\mathbf{U}_\nu, \mathbf{U}_\Lambda \leftarrow 0$
3 $\mathbf{V}_{vv} \leftarrow \mathbf{V}_\mu \mathbf{V}_\mu^\top + \mathbf{V}_\Lambda^{-1}$
4 **foreach** *i in num_threads* **do**
5 *Create thread i. fseek to the part of the data the thread i is working on*
6 **foreach** *w in data* **do**
7 mutex-lock(w)
8 **for** *c in window(w)* **do**
9 $\mathbf{U}_\nu[w] \leftarrow \mathbf{U}_\nu[w] + 0.5 \cdot \mathbf{V}_\mu[c]$
10 $\mathbf{U}_\Lambda[w] \leftarrow \mathbf{U}_\Lambda[w] + 2 \cdot \lambda_{w,c} \cdot \mathbf{V}_{vv}[c]$
11 **foreach** *c in sample(n)* **do**
12 $\mathbf{U}_\nu[w] \leftarrow \mathbf{U}_\nu[w] - 0.5 \cdot \mathbf{V}_\mu[c]$
13 $\mathbf{U}_\Lambda[w] \leftarrow \mathbf{U}_\Lambda[w] + 2 \cdot \lambda_{w,c} \cdot \mathbf{V}_{vv}[c]$
14 mutex-unlock(w)
15 *Destroy thread i.*
16 **foreach** *i in num_threads* **do**
17 **foreach** *w in n* **do**
18 $\mathbf{U}_\Lambda[w] \leftarrow \mathbf{U}_\Lambda[w] + \mathbf{I}$
19 $\mathbf{U}_\mu[w] \leftarrow \mathbf{U}_\Lambda[w]^{-1} \mathbf{U}_\nu[w]$
20 *Similar updates for* $\mathbf{V}$ *. Except, instead of adding* $\mathbf{I}$ at **18** one adds $diag(\boldsymbol{\alpha})$, *the ARD priors.*

the gradient estimate noisier than it ought to be although this is not a serious issue at all. However, since the mutex lock is a trifling price to pay for slightly improved estimates, we stick with it – although an aggressive downsampler might render a mutex lock redundant.

It remains to be argued why Algorithm 4 should converge to a better local optima than Algorithm 3. The simplest explanation is that Algorithm 4 takes into account the entire data set to produce to better estimate of the gradient than the one obtained by Algorithm 3 [HBWP13] in the same amount of wall-clock time. However there exists a much subtler reason for this. The per word forgetting rate scheme

used in Algorithm 3 does not take into account the contexts surrounding the word. Thus, if for instance, a particular set of contexts appear much later in the corpus, the updates reflected on the parameters turn out to be much weaker than what one would desire in such "fresh" set of contexts. Of course, this problem is alleviated to a certain extent by using random mini-batches. Even then it remains starkly inferior to the case where one would use a separate forget rate for each unique (word, context) pair: a rather unrealisable possibility preempted by the space requirements imposed.

6 Experiments and Visualisations

> It's delightful when your
> imaginations come true, isn't it?
>
> *Anne of Green Gables*

In this chapter, we shall attempt to prove that our embeddings are visually and qualitatively sound. Firstly, we will demonstrate that our model indeed works by running it on synthetically manufactured data. Then, in order to visualise the embeddings we make use of t-distributed stochastic nearest neighbour embeddings (t-SNE) [ME08] to reduce the dimensionality of the word embeddings. Next, in order to qualitatively evaluate the embeddings we employ the following tasks:

- Word Similarity Test
- Word Analogy Tasks
- Chunking

Finally, we will also employ the Word Similarity test to conclude that the embeddings produced by Algorithm 3 are generally inferior to the ones generated by Algorithm 4.

6.1 Synthetic Data Set

The synthetic data ought to be so designed as to verify that the model really works. To this end, we had to dispense with the randomly generated negative data stream and instead depend on pre-generated negative stream so as to faithfully predict the expected lower bound of the model.

The data generation is rather straightforward. Data is sampled from the model specified in Figure 6, with the biases ignored. I set the value of the underlying original dimension to $k = 10$. $\mathbf{U}$ and $\mathbf{V}$ are sampled from standard Gaussian distributions. The vocabulary size of the word and its context are assumed without loss of generality to be $m = n = 50$. Thus, the dimensions of $\mathbf{U}$ and $\mathbf{V}$ are $m \times k$ and $n \times k$ respectively. A data sample constitutes a pair x and y, sampled from $[1, m]$ and $[1, n]$ respectively. A sample (x, y) is positive if $\sigma(\mathbf{u}_x^\top \mathbf{v}_y) > 0.5$. Otherwise, it contributes to the negative stream. For the purposes of the experiment, the size

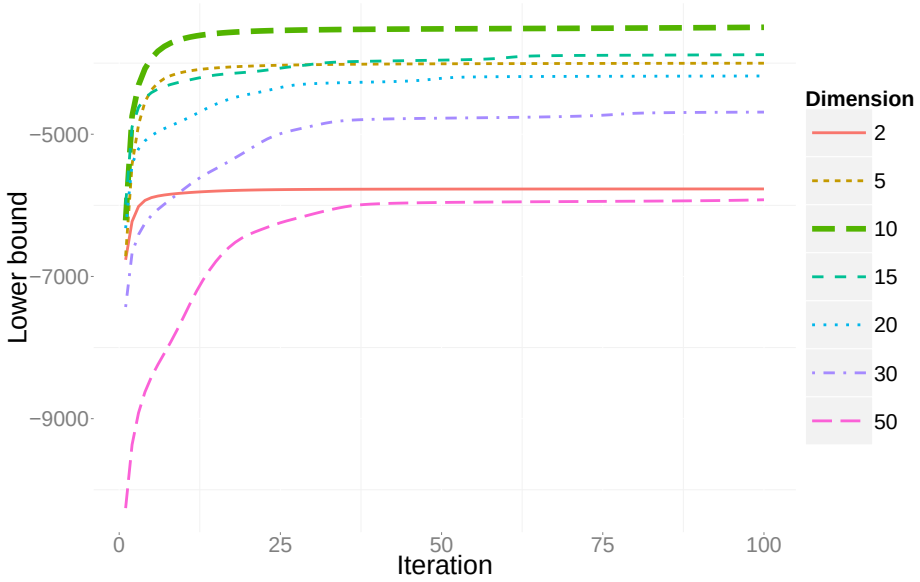


Figure 10: The plot shows the maximisation of the expected lower bound, for the synthetic data, over 100 iterations. The size of the sample space $m = n = 50$. From these sample spaces, a sample data set of size $|\mathcal{D}| = 10000$ is generated from the model specified in Figure 6. The model converges in around 20-25 iterations. The size of the latent vectors, k ranges from 2 to 50 with the optimal lower bound at the true dimension $k = 10$. For $k < 10$, the lower bound remains lower while for $k > 10$, the ARD effectively switches off all extraneous dimensions. All trials took anywhere between 1 and 5 minutes to complete. In all cases where $k > 10$, the model was able to infer the true underlying dimension of the data as approximately 10.

of the data set $\mathcal{D}$ was set to $|\mathcal{D}| = 10000$. The algorithm used here is a single-threaded version of Algorithm 4. I ran the experiment for the latent dimensions $k = 2, 5, 10, 15, 20, 30, 50$. From Figure 10, it is clear that the original dimension $k = 10$ has the best lower bound of all cases. When $k < 10$, even though all the dimensions are utilised, the lower bound is worse than the optimal at $k = 10$. For $k > 10$, the ARD prior effectively switches off the extraneous dimensions leaving only approximately $k = 10$ dimensions 'on'. This is shown using the Hinton diagram of the ARD in Figure 12. The red corresponds to the active components and the green to the components that are switched off. The values for the Hinton diagram are obtained from $\ln\langle\boldsymbol{\alpha}\rangle$, where $\boldsymbol{\alpha}$ is the ARD hyperparameter. Figure 11 compares the lower bounds at the end of the 100^{th} iteration of all the various dimensions tried and proves that the lower bound corresponding to the true dimension is optimal.

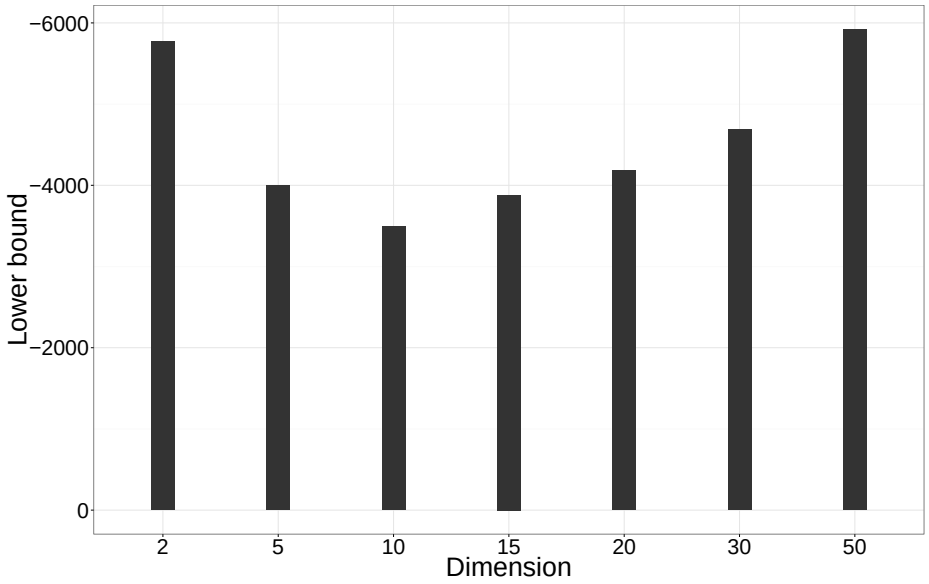


Figure 11: The final lower bound after 50 iterations for each of the dimensions. The lower bound is maximal for the true dimension $k = 10$, while the other dimensions fare worse in comparison. For all $k > 10$ all the extraneous components are switched off allowing only the relevant components to remain active.

6.2 Reuter's Corpus

Reuter's corpus (RCV1) is a collection of Reuter's news stories from the year 2000 for use in research and natural language processing. The corpus is distributed by National Institute of Science and Technology (NIST). The size of RCV1 after the preprocessing steps of removing capitalisation, punctuation and numbers is 987MB. The number of unique tokens with a frequency of at least 10 is 136789. RCV1 has nothing special to recommend it save its traditional use in inducing and evaluating the quality of word embeddings on standard tasks. In order to evaluate our embeddings, we ran both variants of our models on RCV1. Both were run for 200 iterations over a span of 2 days. The dimension of the embedding was set to 50, the minimum frequency to 10, the negative sample size per token to 2, the downsampling rate to $1 \times e^{-5}$ and the window size to 8. In order to evaluate the progress of variational Bayesian methods, one customarily uses the expected lower bound. However, we ran into problems estimating the exact value of ELBO owing to the random negative samples. We therefore instead resorted to using the log likelihood of the data, given the current parameters as a metric to evaluate the progress of the algorithm. This applies for the second variant, Algorithm 4, as shown in Figure 13.

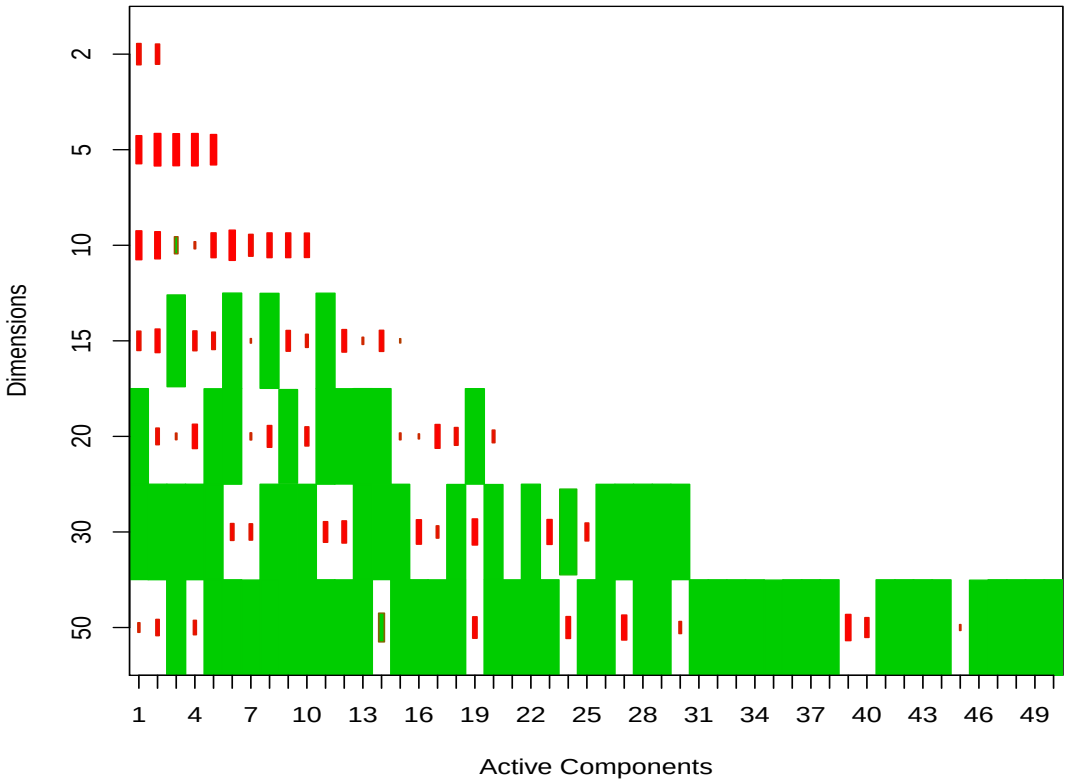


Figure 12: Hinton diagram of the weights given by $\ln\langle\boldsymbol{\alpha}\rangle$. On the x-axis, the number of active components is given. The y-axis corresponds to the dimensions the model is run with, which range from $k = 2$ to 50 with the true dimension at $k = 10$. The red corresponds to the active components of the ARD while the green denotes the components that have been effectively switched off. It is seen that where the number of components exceeds the true dimension of the generating model, most extraneous components are switched off.

As for the SVI variant, Algorithm 3, it is common practice to shy away from measuring the progress of the algorithm with the lower bound and instead use the per-word predictive log likelihood in the case of Latent Dirichlet Allocation [HBWP13] and recall in the case of binary matrix factorisation [HlHG14]. This is because measuring the convergence of SVI algorithms is extremely problematic [Paq14]. One alternative is to evaluate the improvement of the scaled log likelihoods for each batch and plot them after some smoothing. However, I ran into some difficulty doing this since the behaviour of the log likelihood was rather wild. I instead use the word similarity tasks (see Section 6.4.1) to pit Algorithm 3 against Algorithm 4 and

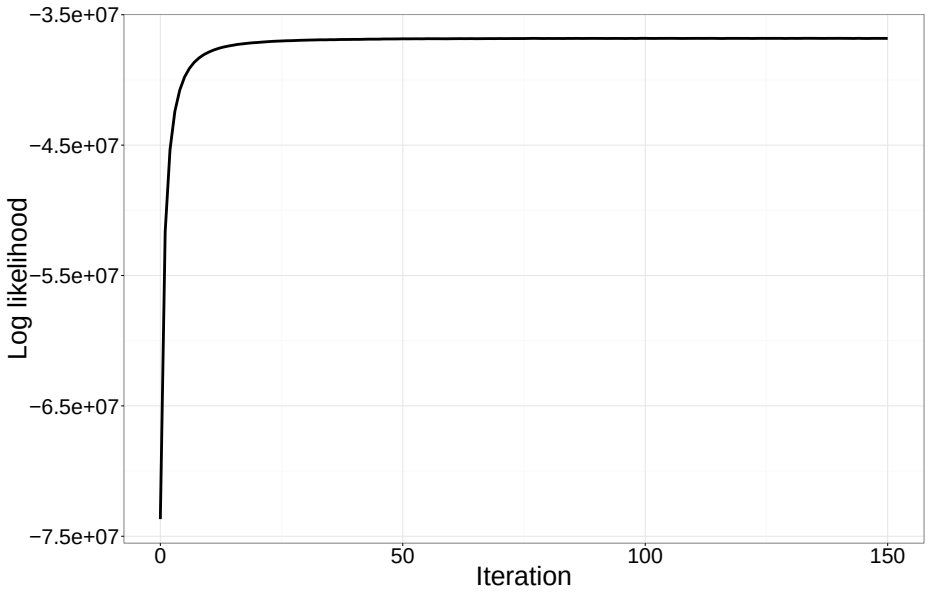


Figure 13: The log likelihood of the RCV1 data, the positive samples alone, plotted against the number of iterations for Algorithm 4. We chose this metric as it proved infeasible to accurately calculate the expected lower bound when taking into account the random negative data samples. The algorithm was run for 200 iterations with a latent dimension of $k = 50$ over a span of 2 days. It however converged to a near optimal result in 20-25 iterations.

demonstrate that the embeddings furnished by the latter prove to be far superior to the former. Thus, for all intents and purposes, when we refer to our 'model' we mean Algorithm 4.

6.3 t-SNE

t-SNE is a nonlinear dimensionality reduction technique developed by [ME08] that lends itself particularly well to visualisation tasks. Unlike typical dimensionality reduction techniques that seek to preserve the global structure, t-SNE constructs two distributions – one over every pair of the original data points such that similar data points have a higher probability of being chosen, and another over every pair of the low-dimensional map – and reduces the KL-divergence between them. Given n k-dimensional vectors, t-SNE calculates the probabilities proportional to the similarities between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, defined as

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}, \qquad p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}$$

where σ_i is defined as the perplexity of the conditional distribution – larger the perplexity less dense the data space. t-SNE learns n d-dimensional vectors $\mathbf{y}_i$, $d < k$. The values of $\mathbf{y_i}$ are determined by minimising the KL divergence of Q with P

$$\mathrm{KL}(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

using gradient descent, where q_{ij} is defined by

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}.$$

This gives a visual output with which to verify our embeddings.

In order to come up with visually pleasing embeddings I clustered the embeddings using k-means with 7000 centres and then picked those words that fell in a few important clusters. The colouring of the words refers to the cluster the word fell in. Note that this colouring is completely unsupervised. We do not have any previous labels to assign the colours. A few such interesting but randomly chosen clusters t-SNE are plotted in the Figures 14, 15 and 16.

6.4 Word Similarity

As the most basic validation of the word embeddings we present the *Word Similarity-353 Test Collection* [FGM+01] a collection of English word pairs as well as corresponding human-assigned similarity judgements. The first set consists of 153 word pairs with similarity scores assigned by 13 subjects while the second set has 200 word pairs with similarity scores assigned by 16 subjects. They were all asked to rate the *relatedness* of a word on a scale of 1 to 10. The collection whose scores reflect the extent of semantic similarity between the word pairs is used to test algorithms implementing semantic similarity measures. [DRFU12] claim that the Spearman's correlation test scores of the similarity scores and the cosine distance between the vectors of the corresponding word pairs to accurately gauge the quality of embeddings. This is shown in Table 2. We also find it necessary to caution that the correlation score is a noisy metric in that while it does measure semantic relatedness, it doesn't reflect the syntactic correlations between words, as we later found out. We note that Algorithm 4 outperforms all the current published word in the word similarity task.

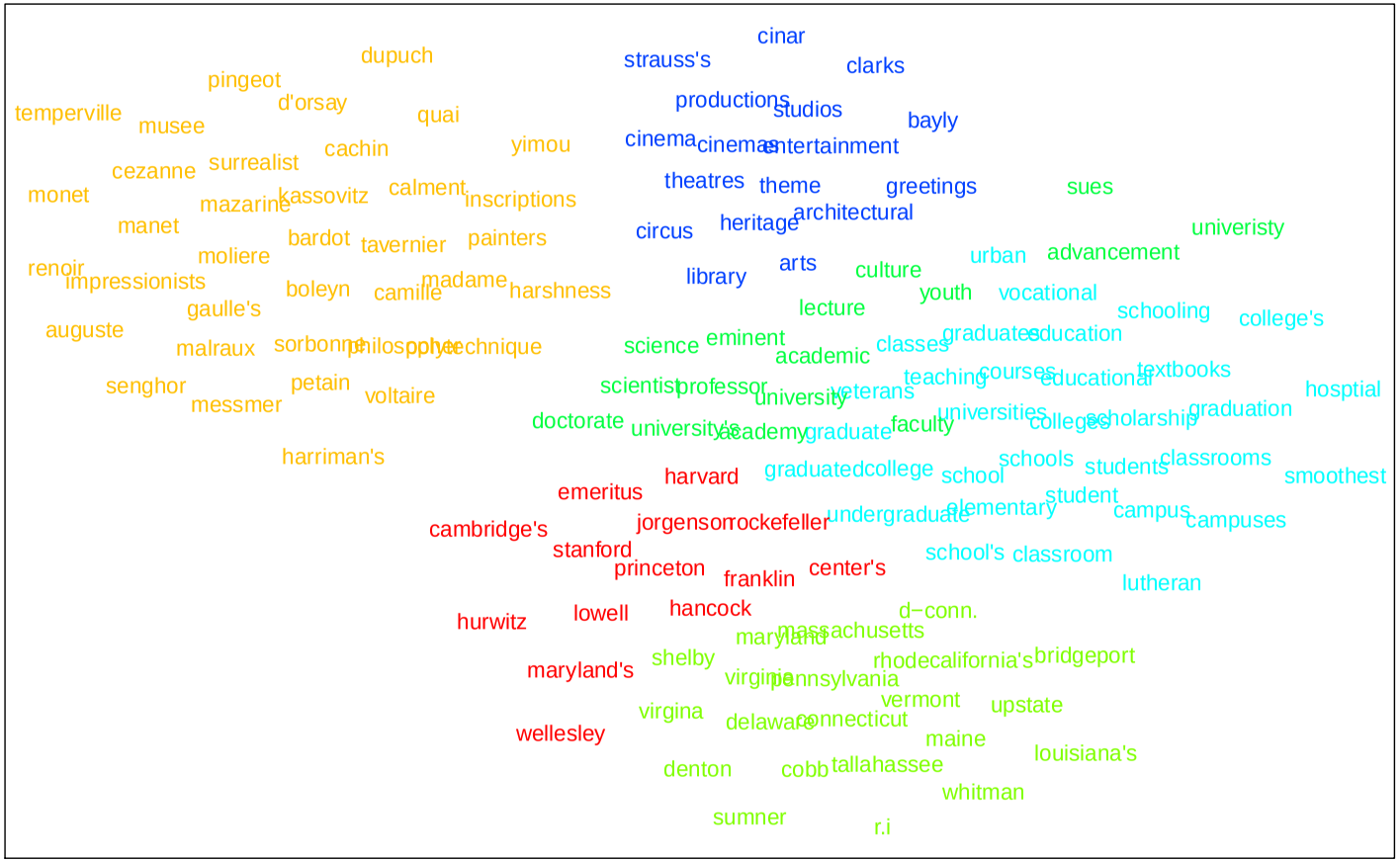


Figure 14: **Universities, arts and culture**: This t-SNE plot shows clusters from academics and arts. To the top left we have a cluster of French universities; to the centre top, we have a cluster on heritage and arts; towards the right, in light blue, we have the cluster for universities and schools and another cluster for students and professors just above. The two clusters at the bottom are in all probability American universities and states mixed together. In keeping with our intuition we find that *Princeton*, *Harvard* and *Stanford* all belong to the same cluster of what are essentially well-known universities.

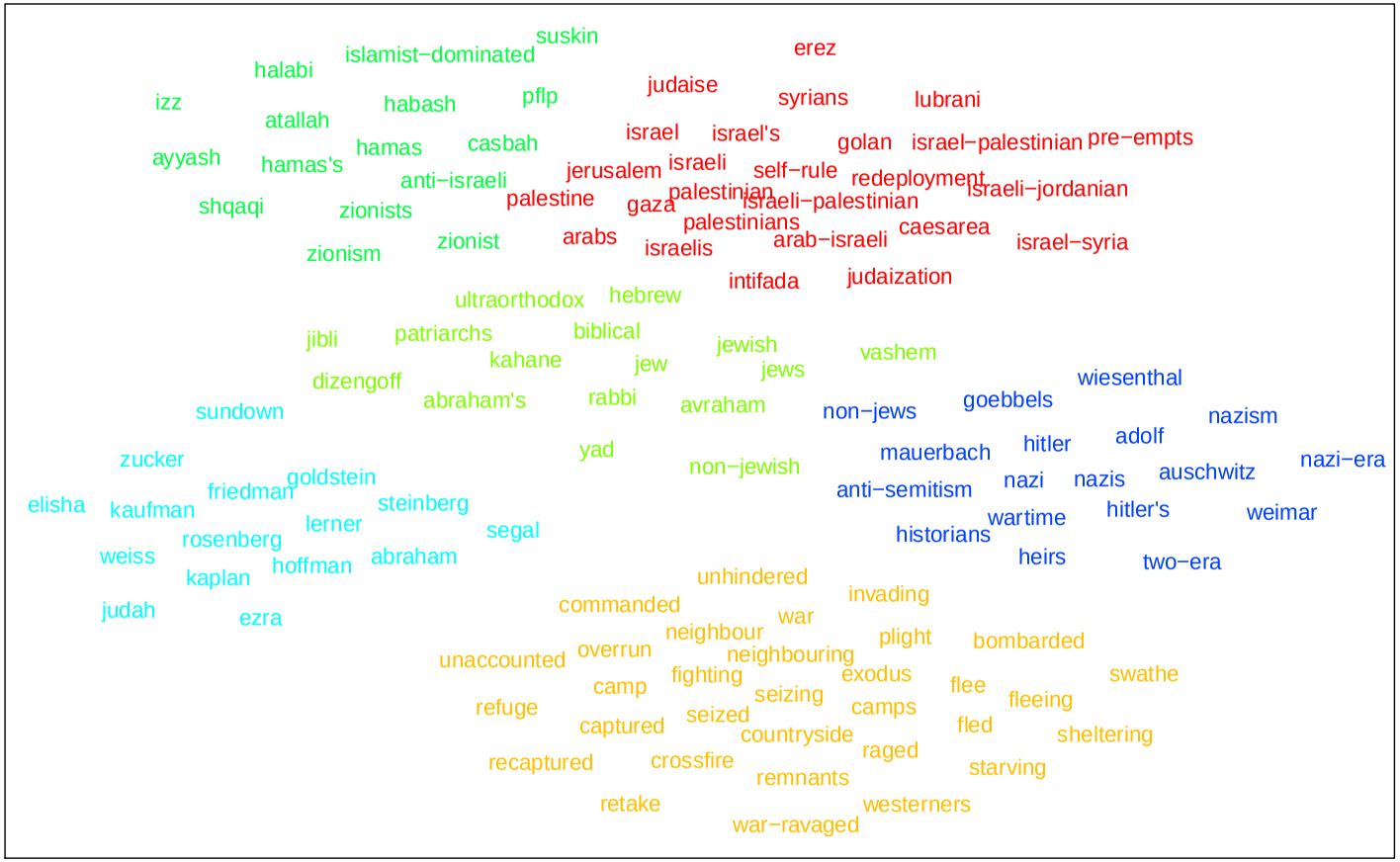


Figure 15: **Jews, Israel and Nazis**: This t-SNE plot shows clusters for excerpts from Jewish history. The green cluster (top left) has some current extremist groups. The red cluster includes the formation of the Jewish state. The dark blue cluster is from the Nazi era, while the light blue one refers to German Jews. The light green cluster is from ancient Jewish history. The cluster at the bottom is about the ravages of war.

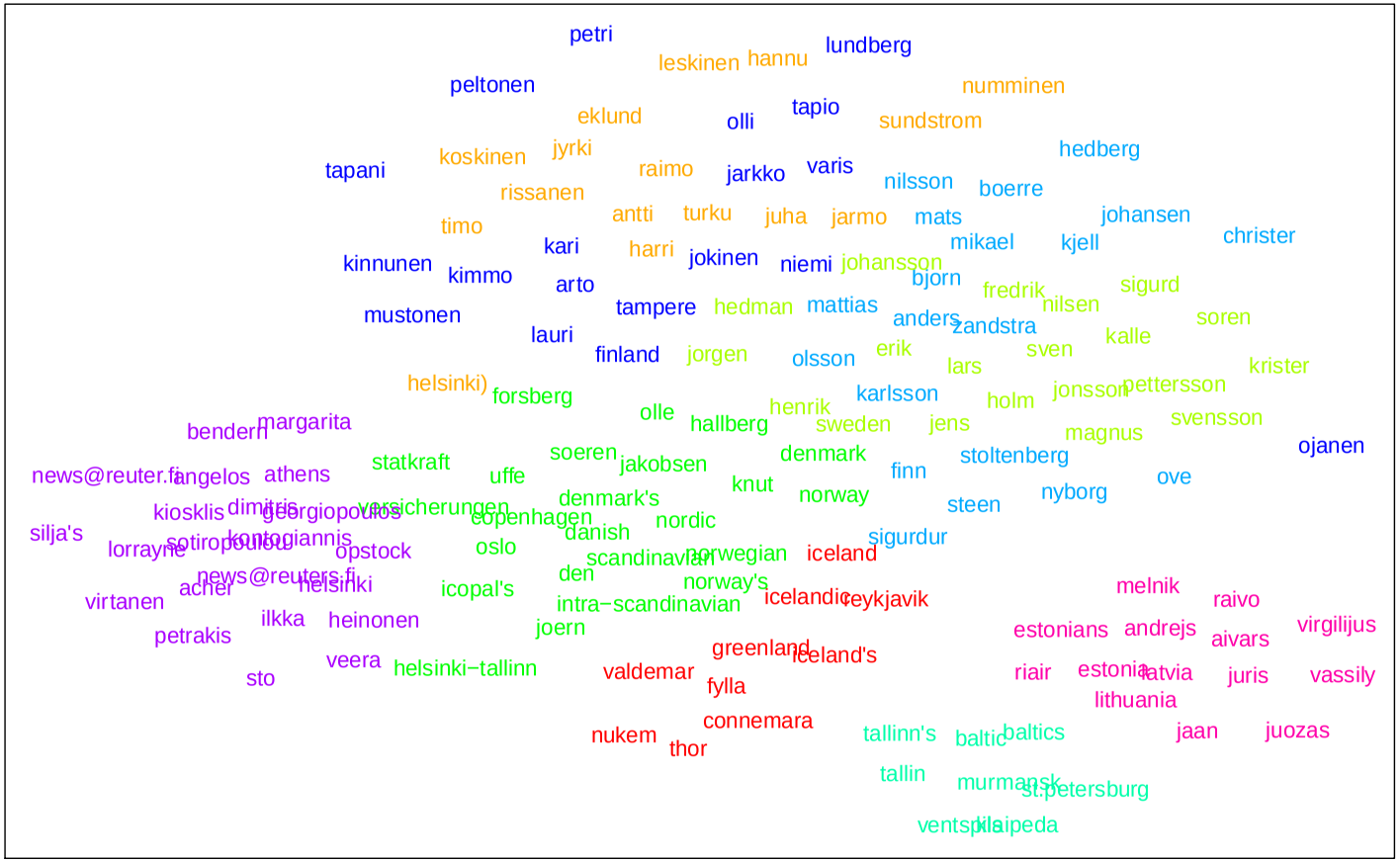


Figure 16: **Scandinavia**: This rather interesting plot clearly shows the geographical and linguistic boundaries of Northern Europe. The pink cluster (bottom right) shows the Baltic states, while the green-blue cluster is a Russo-Baltic cluster. The red cluster is one about Greenland, Iceland and parts of Ireland, the furthest removed from Scandinavia proper. The green cluster (centre-left) is one about Nordic countries specifically. The dark blue and the ochre clusters are essentially Finnish names and cities, which are clearly demarked from Swedish names and cities shown in the light blue and green clusters. The least interesting is the purple cluster about Finnish news agencies. *Ojanen* seems to be an exceptional case.

Word Embeddings Model	Citation	$\rho \times 100$
Algorithm 3		19.47
Turian (CW)	[TRB10]	28.08
PCA	[LL13]	30.25
Turian (HLBL)	[TRB10]	35.24
Eigenwords (LR - MVL(II))	[DFU11]	37.9
word2vec (SK)	[Dhi14]	42.73
word2vec (CB)	[Dhi14]	42.97
Eigenwords (OSCCA)	[DRFU12]	43.00
SENNA	[CWB$^+$11]	44.32
Eigenwords (TSCCA)	[DRFU12]	44.85
Eigenwords (LR - MVL(I))	[DFU11]	43.83
Algorithm 4		**46.86**

Table 2: Word Similarity Test: The Spearman correlation between the cosine similarity of the word embeddings and the human-assigned scores are evaluated. Note that our vectors are 50 dimensional and have been trained on the RCV1 corpus with a window size of 2 and a vocabulary size of 100000. We had to eliminate two entries in the data set corresponding to *(tiger, carnivore)* and *(precedent,antecedent)* since these did not exceed our minimum count stipulation of 10. Clearly, on this metric Algorithm 4 outperforms a wide range of recent works.

6.4.1 Comparison of the Two Algorithm Variants

From Table 2, we also see that the performance of Algorithm 3 is rather poor in comparison both to Algorithm 4 as well as the other models. Thus, in the absence of more reliable benchmarks we tentatively conclude that the performance of Algorithm 4 is far superior to that of Algorithm 3 which is also in keeping with our line of argument we posited in Section 5.3.2. To be fair, the stochastic algorithm incorporates only the most necessary and basic features of the SVI paradigm and enjoys the potential for vast improvement.

6.5 Semantic and Syntactic Word Analogy Tasks

In order to more systematically evaluate the quality of the representations, we use the comprehensive test set of [MCCD13] that consists of five types of semantic

Task	Pair 1		Pair 2	
capital-common-countries	Paris	France	Helsinki	Finland
capital-world	Astana	Kazakhstan	Harare	Zimbabwe
currency	Angola	kwanza	Iran	rial
city-in-state	Chicago	Illinois	Stockton	California
family	brother	sister	father	mother
adjective-to-adverb	apparent	apparently	rapid	rapidly
opposite	easy	hard	legal	illegal
comparative	big	bigger	tall	taller
superlative	easy	easiest	hard	hardest
nationality-adjective	Switzerland	Swiss	Finland	Finn
past-tense	walking	walked	swimming	swam
plural	moose	mooses	goose	geese
plural-verbs	work	works	talk	talks

Table 3: Examples from the analogy type tasks. The top section constitutes the semantic questions while the bottom section constitutes the syntactic questions.

questions and seven types of syntactic questions. Some examples from this test set are shown in Table 3. Previous papers [TRB10] used the word vectors to find semantic relatedness between similar objects. For example, it is not hard to establish the closeness of two countries, say *France* and *Finland*. However, it is a much more complicated task to recognise the relationship between a word pair, say, *France* and *Paris* and use that to correctly predict that *Helsinki* holds a similar relationship to *Finland*. In the case of syntactic analogies, given *easy* and *easiest*, the model should recognise the superlative relationship between these and accurately predict the superlative of *hard* to be *hardest*.

The questions are all framed in this pattern: $A : B :: C : X$, where A and B are the first word pair, and we need to find X that holds the same relationship to C as A does to B. This relationship can be either syntactic or semantic. There are in total 19544 such questions of which 8869 are semantic and the rest syntactic. The questions were formed by taking a fixed number of word pairs and listing a Cartesian product between them. Only single token words were included.

The percentage of semantic relationships accurately predicted for each kind of semantic analogy relationship is shown in Table 4. Similarly, Table 5 shows the results

Semantic Analogies	word2vec	Our model
capital-common-countries	**63.24**	53.75
capital-world	**44.41**	41.20
currency	**39.65**	36.88
city-in-state	**31.58**	27.92
family	**30.86**	26.96

Table 4: *Semantic word analogy results*: We compare our embeddings with the embeddings from word2vec, both trained on the RCV1 corpus with a window size of 8. The results are the percentage of the analogy tasks gotten right. It is clear that our model, though outperformed by word2vec shows competitive results.

Syntactic Analogies	word2vec	Our model
adjective-to-adverb	**11.29**	4.54
opposite	**7.72**	3.43
comparative	**14.61**	6.49
superlative	**13.68**	5.18
present-participle	**14.40**	6.46
nationality-adjective	**25.48**	19.36
past-tense	**23.58**	17.79
plural	**23.65**	17.22
plural-verbs	**22.66**	16.29

Table 5: *Syntactic word analogy results*: We compare our embeddings with the embeddings from from word2vec, both trained on the RCV1 corpus with a window size of 8. The results are the percentage of the analogy tasks that were accurately predicted. However the word2vec representations perform significantly better than ours.

obtained from syntactic analogy tasks. Clearly, our model performs comparably to word2vec in most semantic tasks, although word2vec proves better in all cases. However, as far as syntactic questions are concerned, the word2vec representations beat our model by a significant margin. We will try and analyse the cause of this in Section 7.1

For the sake of fairness, we ought to mention that the word2vec model easily outperforms our model if we were to tweak a few of its parameters. But all factors

Chunker	Features
Baseline	The current word, the two left and right contexts and their corresponding tags
Extended	Baseline + the embeddings corresponding to the words and their contexts.

Table 6: Feature templates used for CRFsuite.

being the same, we can with some confidence assert that our model gives semantic results that are comparable to the word2vec model.

6.6 Chunking

Text chunking consists of splitting text into syntactically coherent groups of words. For example, the sentence

The rain in spain stays mainly in the plain

can be chunked into

[NP : *The rain*][PP : *in Spain*][VP : *stays*][ADVP : *mainly*][PP : *in the plain*]

where NP, PP and ADVP refer to *noun phrases*, *prepositional phrases* and *adverbial phrases*. Such chunks can be viewed, at some level, as intermediaries to full-blown textual parsing.

A commonly used baseline chunker in word embeddings literature is offered by by Sha and Pereira's linear conditional random field shallow parser [SP03]. In keeping with most previous literature on word embeddings we utilise the CRFsuite by Naoaki Okazaki [Oka07] for the chunking task. The features used by the baseline chunker and the extended chunker using the word embeddings are given in Table 6

The data set from CoNLL 2000 shared task for chunking [TKSB00] consists of a training set of 8936 sentences, out of which 1000 sentences are sampled randomly for development in practice. The CRFchunker is trained on the remaining 7936 sentences along with their extended features and the resulting F1-scores are evaluated, as shown in Table 7. It is important to note that the $L2$-regularisation parameter has a lot of effect on the F1-score.

System	Citation	Dev	Test
Baseline	[TKSB00]	94.16	93.79
Our Model		94.26	93.90
word2vec	[MCCD13]	–	94.02
CCA	[DRFU12]	–	**94.23**
Brown clusters	[BPPM93]	**94.67**	94.11

Table 7: F1-scores for the chunking task. While our model managed to beat the baseline scores, we were unable to improve on the F1-scores of already existing models. This is because our model does not capture synactic relationships well enough.

While running the baseline, we obtained the results claimed by [TRB10], not using a L2- regularisation constant of 2 or 3.2, but of 0.05 The reason behind this strange behaviour remains a mystery. While our model did beat the baseline scores in both the development and the test set, we unfortunately failed to improve on the already existing models. We attribute this failure to the inability of our model to capture syntactic relationships.

6.7 Summary

We now summarise the results. We first demonstrated that our model works by running it on a synthetic data set. The results speak for themselves. We then plotted the progress of Algorithm 4, by using the log likelihood instead of the more typical expected lower bound. This is because the existence of random negative samples renders an accurate estimation of the lower bound hideously difficult. However, this too convinces us that our model indeed learns. We then established the effectiveness of our representations using the word similarity tasks, where Algorithm 4 proved to be the best representation. We then used this task, to corroborate – perhaps not too satisfactorily – our initial suspicion that the stochastic variant of our model Algorithm 3 performs poorly. We then used t-sne to reduce the dimensionality of the vectors, to visualise the representation. A simple clustering of the embeddings and selecting a few choice clusters evidenced that our intuition was correct. We also showed how our model shows results comparable to word2vec in semantic word analogy tasks, while performing rather badly in syntactic analogy tasks. Clearly, this latter aspect of our model needs to be rectified. Finally, in keeping with most word

embeddings literature, we tested our representations on chunking tasks. Although our representations did beat the baseline scores, we found that it performed worse than some of the other more popular models. However, the margin of difference between the state-of-the-art and the baseline is rather minuscule and the F1-scores are not too significant.

7 Conclusion

> The end of all our exploring will be
> to arrive where we started and
> know the place for the first time
>
> *T. S. Eliot*

In the preceding chapters we saw how word embeddings of reasonable quality could be induced with a fully Bayesian model. Before recapitulating we will briefly address the deficiencies of our model and hypothesise as to their causes.

7.1 Retrospection

Why did the model do relatively well on semantic tasks but not so much so on syntactic tasks? It is hard to say. My first conjecture was that this behaviour had something to with the window size. The words within a small window provide semantic information, while increasing the size helps account for the syntactic information. However, the word2vec runs with the window size of 2, producing much better syntactic results than our own model. Arguably, their downsampler does a much better job of increasing the effective window size by discarding frequent contexts.

My second line of argument is that the model learns what I shall call *centroid*-based vectors, where the vectors agglomerate strongly around a mean. This agrees with the characteristics of the Bayesian model with a mean square error risk function where the mean of the posterior minimises the mean square error and the Bayes risk is given by its variance. For example, while the vectors for *king* and *queen*, and *man* and *woman* would cluster separately using k-means, simple vector operations such as

$$\text{vec}(king) - \text{vec}(queen) + \text{vec}(man) \approx \text{vec}(woman)$$

would fail to hold. If anything, this portends appalling results for syntactic tests. Alternatively, we can view the vectors as not being *pushed* far apart enough to allow the vector operations to be of any account. My speculation is that this is because of our negative samples being drawn from a uniform distribution instead of a categorical distribution such as the one used in word2vec. While this might prove to be the reason, we have, in our defence, evidence from previous literature

[PKW14] that uniform negative samples give an estimate superior to frequency-based selection. We conclude that our model effectively learns semantic information as evinced strongly in our experiments.

7.2 Recapitulation

The idea that words are recognised by the the contexts in which they occur languished in desuetude until processor evolution allowed us to put it to good use. The earliest word representations were induced with n-gram cluster-based representation [BdM+92] and global representation schemes such as latent semantic analysis [LD97]. Unlike these, distributional representations employ immediate word contexts to generate dense and real-valued representation. This thesis attempted to bring about a convergence of two such prominent works, namely the word2vec [MCCD13], a shallow neural network model employing negative sampling and the multi-view CCA style embeddings of [DRFU12]. These both attempt to solve the same problem: factorising a word co-occurrence matrix into two or more matrices.

In our thesis, apart from crystallising the best ideas of both, we also introduced a completely Bayesian, stochastic variational algorithm that potentially scales to any length of data. The third contribution was that while all previous literature produced point estimates of the embeddings, our model predicts continuous Gaussian densities in the latent space. Such a model, as far as we know, is the first to predict such densities as part of its inference process. Clearly, this has its drawbacks in that it increases the inference time polynomially in proportion to the size of our latent embeddings. For example a latent embedding of dimension k would require a covariance of $\frac{k(k+1)}{2}$. While we were not able to convincingly demonstrate the use of such covariances, we surmise that such covariances could be potentially used to mitigate the problems mentioned in Section 7.1. The usefulness of the ARD priors fades away when we limit the size of k to small values. An alternative is to create a global co-occurrence matrix [LC14] for the entire corpus and run the model on this. This should reduce the computation time to the order of a few hours as well as do away with the need for negative sampling (resulting in a calculable lower bounds) and will be the subject of future investigation.

The stochastic version, with its noisy likelihood, provides a good approximation, but a still better one can be obtained by modifying it into a scalable version for finite data. We employ a multithreaded model where each thread locally calculates noisy estimates of likelihoods with downsampling in the part of the data it is working on

and combines the estimates to obtain a better approximation of the likelihood than one obtained by a purely stochastic version. This has provided demonstrably better results.

Finally we trained the modified multithreaded algorithm on the Reuters' Newswire Corpus, a 1.2 gigabyte textual corpus of news data from the years 1996–1997. We let the algorithm run for 200 iterations, with a context window of 2 over a span of one and a half days and used the resultant embeddings for all of our evaluation tasks. We resorted to the t-distributed stochastic neighbour embedding as visual sanity tests.

As a quantitative sanity check, we used the WordSim-353 data set and calculated the Spearman's correlation scores to support our claim that the distance between the vectors corresponded to the human similarity scores. Because our model had a strong tendency to learn semantic properties, we obtained the highest score among all embeddings.

However, we did not beat the state-of-the-art embeddings in the semantic and syntactic analogy and chunking tasks. We added the embeddings as feature vectors to CoNLL chunking challenge, hoping to improve upon the baseline scores. Unfortunately, the improvement was not remarkable – owing to poor syntactic discrimination.

8 Future Work

> Because when you are imagining,
> you might as well imagine
> something worth while.
>
> *Anne of Green Gables*

It is common practice to leave this section as a footnote to the conclusion. But we realise the work we have done is woefully inadequate and only the first step in what we hope shall be a fully Bayesian approach to unsupervised natural language processing. In the following subsections we look and try to formalise briefly a few directions of progress.

8.1 Direct Extensions

One of the prominent, albeit underutilised, contributions of this thesis is the representation of the latent dimensions as Gaussian densities instead of the more typical point estimates. The covariance of the Gaussian densities allows us to take the uncertainty of each dimension into account. For instance, we earlier used the Euclidean distance between the Gaussian means for the WordSim-353 [FGM+01] which gave very good results. An even better alternative would be to use the probability product kernel [JKH04] of the Gaussian densities as a measure of word similarity. Given the Gaussian densities corresponding to two words $p(w) \sim \mathcal{N}(\mu_w, \Sigma_w)$ and $p(c) \sim \mathcal{N}(\mu_c, \Sigma_c)$ the probability product kernel or the inner product of two Gaussians is again a Gaussian given by [VM14]:

$$\int \mathcal{N}(x; \mu_w, \Sigma_w)\mathcal{N}(x; \mu_c, \Sigma_c)\ \mathrm{d}x = \mathcal{N}(0; \mu_w - \mu_c, \Sigma_w + \Sigma_c).$$

This result directly follows from the convolution property of Gaussians – the convolution of two Gaussians is a Gaussian.

Another possibility is to investigate the effective use of the ARD prior. In our experiments, we had to limit the latent dimensionality to $k = 50$ because of the constraints imposed by the dimensionality of the covariance matrix $\frac{k(k+1)}{2}$ and by the large vocabulary size. Since there were no extraneous dimensions, the working of the ARD could not be properly demonstrated. By limiting the vocabulary size to the n most frequent words (usually in the order of 10^3) along the lines of [LL13], we can run our model over a much larger latent dimension k and let the ARD choose the

dimension best suited. Finally, to improve the performance of the stochastic variant of our algorithm we could resort to more careful step-size selection as demonstrated by [RWBX13].

In the sections that follow, I will outline a few ideas that are only tangential extrapolations of our model, but nevertheless enjoy immense practical benefits.

8.2 Unsupervised POS Tagging

Unsupervised POS tagging is simply the end of a long list of research on word clusters by [BdM$^+$92] and [CWB$^+$11] *inter alia*. [GG07] discusses a fully unsupervised Bayesian approach involving a simple trigram based POS tagger defined as,

$$t_i|t_{i-1} = t, t_{i-2} = t', \tau^{(t,t')} \sim \text{Mult}(\tau^{(t,t')})$$
$$w_i|t_i = t, \omega^t \sim \text{Mult}(\omega^t)$$

with symmetric Dirichlet priors for ω^t and $\tau^{(t,t')}$. By integrating over ω^t and $\tau^{(t,t')}$ they present a collapsed Gibbs' sampler for inference and reportedly achieved a tagging accuracy of 86.8%. It is not hard to see the striking resemblance this bears with Latent Dirichlet Allocation (LDA) [Ble12]. To begin with, we could posit a simple ensemble learning method that uses many-to-one POS mapping learnt from the word clusters obtained from our model in conjunction with [GG07] for better predictive performance. A bolder attempt involves incorporating [GG07] as a cog in our Bayesian machinery. Incidentally, adapting non-parametric models, such as the Dirichlet process, allows for modelling language independent POS taggers [VGVG09].

8.3 Bayesian Grammar Induction

[KS06] presents a variational Bayesian grammar induction algorithm. Consider a Probabilistic Context Free Grammar (PCFG) tuple $G = (V_N, V_T, R, S, \boldsymbol{\theta})$, constituting the non-terminals, the terminals, the production rules, the start symbol and the probabilities of the production rules respectively. For any $A \in V_N$, $a \in (V_N \cup V_T)^+$ the corresponding production rule $r : A \to a \in R$, has probability $\theta_A(a)$. Clearly, $\boldsymbol{\theta}_A$ is Dirichlet distributed with hyperparameters $\mathbf{u}_a$, and $\boldsymbol{\theta}$ is a product of such Dirichlet distributions. [Kur04] outlines a variational approximation for estimating the posteriors of $\mathbf{u}$. Grammar induction proceeds roughly as follows:

1. Begin with an initial grammar and estimate the posterior of $\mathbf{u}$.
2. While the free energy decreases
   (a) Split the i^{th} rule and re-estimate $\mathbf{u}$.
   (b) Delete production rules while free energy decreases
   (c) If there is no decrease in free energy, quit.
3. While the free energy decreases
   (a) Merge rules with highest cosine similarity between their parameters $\boldsymbol{\theta}_i$ and re-estimate $\mathbf{u}$.
   (b) Delete production rules while free energy decreases
   (c) If there is no decrease in free energy, quit.
4. Output the induced grammar.

[KS06] reports that the Bayesian approach to grammar induction showed better results than earlier ones using the Inside-Outside algorithm [SRO93] or its modifications [HM98] for induction. Let us now discuss the dependency grammar as a special case of grammar induction.

8.3.1 Dependency Grammar Induction

[KM04] introduced a modified unsupervised dependency grammar induction that employed part-of-speech tags as part of its generative story to model its parameters. [SACJ11] demonstrated that Brown clusters and used as word classes for dependency grammar induction *surpassed* the performance of gold part-of-speech tags. They performed a number of experiments involving different class assignment schemes such as one-to-one, union-all and most frequent tags among others. They then study the effects of hierarchical and flat Brown clusters go on to conclude that the drawback of word embeddings is not so much the lack of supervision as the the prevalence of the *one class per word* schemes, that is, the inability of word clusters to represent polysemous words. We hypothesise that a unified Bayesian model inducing word classes and a dependency grammar iteratively should show improved results to the two-step pipeline introduced by [SACJ11]. A good while ago, [WSM08] presented a Pitman-Yor process Bayesian model for dependency parsing. Incorporating this into our existing model, while perhaps requiring some effort, seems possible. One could also come up with cyclic models that alternate between improving induced

grammars from the current embeddings and using the dependency parses of the current grammar as contexts for the subsequent embeddings.

8.4 A Unified Bayesian Model for Unsupervised NLP

We draw inspiration from [CWB+11] to speculate upon a Bayesian model that performs unsupervised learning of four classical NLP tasks, viz,

1. POS tagging
2. Chunking
3. Parsing
4. Semantic Role Labeling (SRL)

The first three items are clearly incremental in nature, and we could form a simple Bayesian pipeline. SRL might prove harder, but we think that we could use Markov Logic Networks [RD06] in conjunction with the embeddings induced to reason probabilistically about thematic relations.

Will a computer ever be able completely draw valid rules and patterns from natural language on its own? Is it fair that we model something so monstrous on our subjective whims? Should we try and mimic the way humans learn language[9], or are we better off creating statistical models that are better understood? These questions continue to haunt us and time alone will reveal whether the structure intrinsic to natural language can ever be *completely* disentangled from all that seeming chaos.

[9]The neural network community didn't get very far modelling the human brain.

References

Ama98 Amari, S.-I., Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10,2(1998), pages 251–276.

Att99 Attias, H., Inferring parameters and structure of latent variable models by variational bayes. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, San Francisco, CA, USA, 1999, Morgan Kaufmann Publishers Inc., pages 21–30.

BdM+92 Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D. and Lai, J. C., Class-based N-gram Models of Natural Language. *Comput. Linguist.*, 18,4(1992), pages 467–479.

Bea03 Beal, M. J., *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

Bis99 Bishop, C. M., Variational Principal Components. *Proceedings Ninth International Conference on Artificial Neural Networks, ICANN'99*, volume 1. IEE, January 1999, pages 509–514.

Bis06 Bishop, C. M., *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

BJ06 Bach, F. R. and Jordan, M. I., A Probabilistic Interpretation of Canonical Correlation Analysis. *Technical Report 688, Department of Statistics*.

Ble12 Blei, D. M., Probabilistic Topic Models. *Communications of the Association for Computing Machinery*, 55,4(2012), pages 77–84.

Boh92 Bohning, D., Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, 44, pages 197–200.

BPPM93 Brown, P. F., Pietra, V. J. D., Pietra, S. A. D. and Mercer, R. L., The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19,2(1993), pages 263–311.

BS94 Bernardo, J. M. and Smith, A. F. M., *Bayesian Theory*. John Wiley & Sons, New York, 1994.

CWB+11 Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P., Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12, pages 2493–2537.

DFU11 Dhillon, P. S., Foster, D. P. and Ungar, L. H., Multi-View Learning of Word Embeddings via CCA. *Neural Information Processing Systems (NIPS)*, Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F. C. N. and Weinberger, K. Q., editors, 2011, pages 199–207.

Dhi14 Dhillon, P. S., *Advances in Spectral Learning with Applications to Text Analysis and Brain Imaging*. 2014.

DRFU12 Dhillon, P. S., Rodu, J., Foster, D. P. and Ungar, L. H., Two Step CCA: A new spectral method for estimating vector models of words. *Proceedings of the 29th International Conference on Machine learning*, ICML'12, 2012.

FGM+01 Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G. and Ruppin, E., Placing search in context: The concept revisited. *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, New York, NY, USA, 2001, ACM, pages 406–414.

GDMB08 González, I., Déjean, S., Martin, P. G. P. and Baccini, A., Cca: An r package to extend canonical correlation analysis. *Journal of Statistical Software*, 23,12(2008), pages 1–14.

GG07 Goldwater, S. and Griffiths, T. L., A fully Bayesian approach to unsupervised part-of-speech tagging. *ACL*, Carroll, J. A., van den Bosch, A. and Zaenen, A., editors. The Association for Computational Linguistics, 2007.

GH12 Gutmann, M. and Hyvärinen, A., Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *Journal of Machine Learning Research*, 13, pages 307–361.

GL14 Goldberg, Y. and Levy, O., word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *Computing Research Repository*, abs/1402.3722.

HBWP13 Hoffman, M. D., Blei, D. M., Wang, C. and Paisley, J., Stochastic Variational Inference. *Journal of Machine Learning Research*, 14,1(2013), pages 1303–1347.

HlHG14 Hernandez-lobato, J. M., Houlsby, N. and Ghahramani, Z., Stochastic Inference for Scalable Probabilistic Modeling of Binary Matrices. *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. JMLR Workshop and Conference Proceedings, 2014, pages 379–387.

HM98 Hogenhout, W. R. and Matsumoto, Y., A fast method for statistical grammar induction. *Natural Language Engineering*, 4,3(1998), pages 191–209.

HNP09 Halevy, A., Norvig, P. and Pereira, F., The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, 24,2(2009), pages 8–12.

HTRK08 Honkela, A., Tornio, M., Raiko, T. and Karhunen, J., Natural Conjugate Gradient in Variational Inference. *In Proceedings of the 14th International Conference on Neural Information Processing (ICONIP 2007)*, volume 4985 of *Lecture Notes in Computer Science*. Springer-Verlag, 2008, pages 305–314.

JGJS99 Jordan, M. I., Ghahramani, Z., Jaakkola, T. S. and Saul, L. K., An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37,2(1999), pages 183–233.

JKH04 Jebara, T., Kondor, R. and Howard, A., Probability product kernels. *Journal of Machine Learning Research*, 5, pages 819–844.

Kas98 Kaski, S., Dimensionality Reduction by Random Mapping: Fast Similarity Computation for Clustering, 1998.

Kla14 Klami, A., Polya-Gamma augmentations for factor models. volume Proceedings of the 6th Asian Conference on Machine Learning of *29*. JMLR: Workshop and Conference Proceedings, 2014, pages 112–128.

KM04 Klein, D. and Manning, C. D., Corpus-based induction of syntactic structure: Models of dependency and constituency. *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*,

ACL '04, Stroudsburg, PA, USA, 2004, Association for Computational Linguistics.

Koe10 Koehn, P., *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, first edition, 2010.

KS06 Kurihara, K. and Sato, T., Variational Bayesian grammar induction for natural language. *In International Colloquium on Grammatical Inference*, 2006, pages 84–96.

Kur04 Kurihara, K., An application of the variational Bayesian approach to probabilistic contextfree grammars. *In International Joint Conference on Natural Language Processing Workshop Beyond Shallow Analyses*, 2004.

KVK13 Klami, A., Virtanen, S. and Kaski, S., Bayesian Canonical correlation analysis. *Journal of Machine Learning Research*, 14,1(2013), pages 965–1003.

LB96 Lund, K. and Burgess, C., Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, 28,2(1996), pages 203–208.

LC14 Lebret, R. and Collobert, R., Rehabilitation of count-based models for word vector representations. *CoRR*, abs/1412.4930.

LD97 Landauer, T. K. and Dutnais, S. T., A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *PSYCHOLOGICAL REVIEW*, 104,2(1997), pages 211–240.

Lia05 Liang, P., Semi-supervised learning for natural language. 2005.

LL13 Lebret, R. and Lebret, R., Word Emdeddings through Hellinger PCA. *Computing Research Repository*, abs/1312.5542.

MCCD13 Mikolov, T., Chen, K., Corrado, G. and Dean, J., Efficient estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781.*

ME08 Maaten, L. v. d. and E., H. G., Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research*, 9, pages 2579–2605.

MH07 Mnih, A. and Hinton, G., Three New Graphical Models for Statistical Language Modelling. *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, New York, NY, USA, 2007, pages 641–648.

MU05 Mitzenmacher, M. and Upfal, E., *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

Oka07 Okazaki, N., Crfsuite: a fast implementation of conditional random fields (crfs), 2007.

Paq14 Paquet, U., On the Convergence of Stochastic Variational Inference in Bayesian Networks. *Neural Information Processing Systems (NIPS) 2014, Workshop on Advances in Variational Inference*, 2014.

Par88 Parisi, G., *Statistical field theory*. Frontiers in Physics. Addison-Wesley, Redwood City, CA, 1988.

PKW14 Paquet, U., Koenigstein, N. and Winther, O., Scalable Bayesian Modelling of Paired Symbols. *pre-print*.

PSW13 Polson, N. G., Scott, J. G. and Windle, J., Bayesian inference for logistic models using Polya-Gamma latent variables. *arXiv:1205.0310*.

RD06 Richardson, M. and Domingos, P., Markov Logic Networks. *Machine Learning*, 62,1-2(2006), pages 107–136.

RJ93 Rabiner, L. and Juang, B.-H., *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

RM51 Robbins, H. and Monro, S., A stochastic approximation method. *Annals of Mathematical Statistics*, 22, pages 400–407.

RS10 Rabiner, L. and Schafer, R., *Theory and Applications of Digital Speech Processing*. Prentice Hall Press, Upper Saddle River, NJ, USA, first edition, 2010.

RV97 Resnick, P. and Varian, H. R., Recommender systems. *Communications of the Association for Computing Machinery*, 40,3(1997), pages 56–58.

RWBX13 Ranganath, R., Wang, C., Blei, D. M. and Xing, E. P., An adaptive learning rate for stochastic variational inference. *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, 2013, pages 298–306.

SACJ11 Spitkovsky, V., Alshawi, H., Chang, A. and Jurafsky, D., Unsupervised dependency parsing without gold part-of-speech tags. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, pages 1281–1290.

SM08 Salakhutdinov, R. and Mnih, A., Bayesian probabilistic matrix factorization using markov chain monte carlo. *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, New York, NY, USA, 2008, ACM, pages 880–887.

SP03 Sha, F. and Pereira, F., Shallow Parsing with Conditional Random Fields. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, Stroudsburg, PA, USA, 2003, Association for Computational Linguistics, pages 134–141.

SRO93 Schabes, Y., Roth, M. and Osborne, R., Parsing the Wall Street Journal with the Inside-Outside Algorithm. 1993, pages 341–347.

TB99 Tipping, M. E. and Bishop, C. M., Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society, Series B*, 61, pages 611–622.

TKSB00 Tjong Kim Sang, E. F. and Buchholz, S., Introduction to the CoNLL-2000 Shared Task: Chunking. *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, ConLL '00, Stroudsburg, PA, USA, 2000, Association for Computational Linguistics, pages 127–132.

TRB10 Turian, J., Ratinov, L. and Bengio, Y., Word Representations: A Simple and General Method for Semi-supervised Learning. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Stroudsburg, PA, USA, 2010, Association for Computational Linguistics, pages 384–394.

VGVG09 Van Gael, J., Vlachos, A. and Ghahramani, Z., The infinite HMM for unsupervised PoS tagging. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Singapore, August 2009, Association for Computational Linguistics, pages 678–687.

VH05 Väyrynen, J. J. and Honkela, T., Comparison of Independent Component Analysis and Singular Value Decomposition in word context analysis, 2005.

VKKK14 Virtanen, S., Klami, A., Khan, S. A. and Kaski, S., Bayesian Group Factor Analysis. *IEEE Transactions on Neural Networks and Learning Systems*, PP,99(2014).

VM14 Vilnis, L. and McCallum, A., Word representations via gaussian embedding. *Computing Research Repository*, abs/1412.6623.

WSM08 Wallach, H., Sutton, C. and Mccallum, A., Bayesian modeling of dependency trees using hierarchical Pitman-Yor priors. *In Workshop on Prior Knowledge for Text and Language, 15–20*, 2008.

ZLDZ07 Zhang, Z., Li, T., Ding, C. and Zhang, X., Binary matrix factorization with applications. *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ICDM '07, Washington, DC, USA, 2007, IEEE Computer Society, pages 391–400.