WELDA: Enhancing Topic Models by Incorporating Local Word Context

Stefan Bunk
Hasso Plattner Institute
Potsdam, Germany
stefan.bunk@student.hpi.uni-potsdam.de

Ralf Krestel
Hasso Plattner Institute
Potsdam, Germany
ralf.krestel@hpi.uni-potsdam.de

ABSTRACT

The distributional hypothesis states that similar words tend to have similar contexts in which they occur. Word embedding models exploit this hypothesis by learning word vectors based on the local context of words. Probabilistic topic models on the other hand utilize word co-occurrences across documents to identify topically related words. Due to their complementary nature, these models define different notions of word similarity, which, when combined, can produce better topical representations.

In this paper we propose WELDA, a new type of topic model, which combines word embeddings (WE) with latent Dirichlet allocation (LDA) to improve topic quality. We achieve this by estimating topic distributions in the word embedding space and exchanging selected topic words via Gibbs sampling from this space. We present an extensive evaluation showing that WELDA cuts runtime by at least 30% while outperforming other combined approaches with respect to topic coherence and for solving word intrusion tasks.

CCS CONCEPTS

• **Information systems** → **Document topic models**; **Document collection models**; • **Applied computing** → *Document analysis*;

KEYWORDS

topic models, word embeddings, document representations

ACM Reference format:
Stefan Bunk and Ralf Krestel. 2018. WELDA: Enhancing Topic Models by Incorporating Local Word Context. In *Proceedings of The 18th ACM/IEEE Joint Conference on Digital Libraries, Fort Worth, TX, USA, June 3–7, 2018 (JCDL '18),* 10 pages.
https://doi.org/10.1145/3197026.3197043

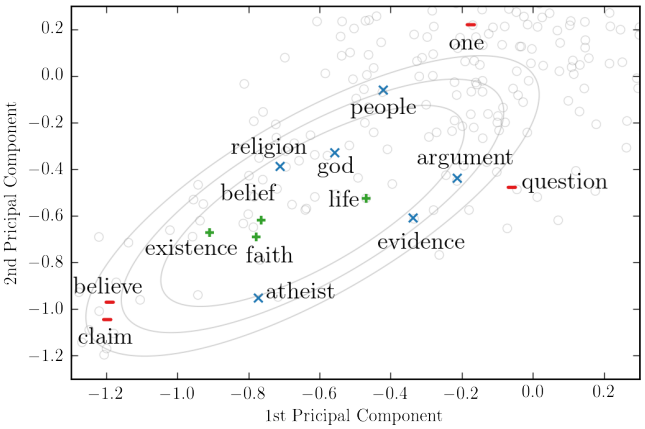


Figure 1: Illustration of an LDA topic in the word embedding space. The gray isolines show the learned probability distribution for the topic. Exchanging topic words having low probability in the embedding space (red minuses) with high probability words (green pluses), leads to a superior LDA topic.

1 INTRODUCTION

Many downstream NLP components benefit from a solid representation of documents and words, for example text categorization, part-of-speech tagging, or machine translation. One popular concept to quantify the meaning of a word is to look at the contexts in which it appears. A famous quote by Firth says: "You shall know a word by the company it keeps" [11]. In other words, if two words occur in similar contexts, they are similar. This assumption, also known as the distributional hypothesis, has driven the development of many models for text representation.

Two of these models are *word embeddings* and *topic models*. Topic modeling tries to reconstruct topics through a generative model from a collection of documents. Each document can then be represented by the topics it covers and each topic can be represented by a set of words. One of the first topic models was latent Dirichlet allocation (LDA) [7]. It models topics as probability distributions over words which makes the topics and the generative process easily interpretable by humans.

The idea of word embeddings is to assign each word in a vocabulary to a real vector in a high-dimensional vector space. This vector representation of the word is called the embedding of the word. Typically, this vector space contains 50 to 600 dimensions [18]. Although the idea is quite old [5, 24], the method has recently gained a lot of popularity in the research community [14, 15, 20]. Open-source tools such as word2vec [1] allow training on billions of words in a supervised manner. Word embeddings are a by-product, namely the internal learned weights of a neural network. Given a text, the neural network learns to predict for each word the sourrounding words. During training, a sliding window runs over the text and for

[1] https://code.google.com/archive/p/word2vec/

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
JCDL '18, June 3–7, 2018, Fort Worth, TX, USA
© 2018 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN 978-1-4503-5178-2/18/06…$15.00
https://doi.org/10.1145/3197026.3197043

each word the neural net adapts its weights to learn to predict its surrounding words. In this way, the context of a word is encoded in the learned weights of the network which then serve as vector representations for the input words.

Word embeddings and topic modeling have different origins. While the former have their roots in the neural network and deep learning community, the latter stems from Bayesian statistics which is why traditionally there is not much research combining the two methods. In this paper, we aim to explore potential synergies between these technologies. Combining the two methods seems promising due to several reasons:

(1) the two models capture different notions of similarity: the word-based embedding models focus on semantic similarity, e.g., "lecturer" and "teacher", while the document-based topic models capture relatedness, e.g., "teacher" and "school".
(2) word embedding models can more efficiently be trained on much larger corpora than standard topic models, such as LDA, thereby allowing more semantic information to be used during the topic model inference step.
(3) out-of-vocabulary (OOV) words are a problem for topic models where unseen words in new documents are typically ignored. Word embedding models do not face the same OOV problems since they are typically trained on billions of words from all domains and can incorporate new terms rather easily, while topic models are typically used to analyse domain-specific, clearly defined collections.
(4) word embeddings lack human interpretability which is where the probabilistic view of topic models can help and facilitate interpretation.

In particular, it might be useful to investigate Baroni's question whether the errors of each model type are complementary [3].

In general, there are two ways of combining these two methods: one can try to improve topic models or one can try to improve word embeddings. We focus on the former and illustrate in Figure 1 how word embeddings can help to improve topic models in our approach. The figure shows an area of a word embedding space projected two dimensions via principal component analysis (PCA). The words marked by black crosses and red minuses represent the top ten words of a topic learned by LDA. The gray dots in the background represent other words in the embedding space. When we plot the learned probability distribution for the words in the embedding space (gray isolines), we see that the words with the lowest probabilities are also the words that fit the least into the topic (the words "one", "question", "claim", and "believe"). The topic could be improved by picking words which are close in the embedding space to the existing words in the topic. In the figure, we illustrated this with the words "existence", "faith", "belief" and "life" which form a stronger topic with the rest of the words.

Further, our WELDA model can handle words that the topic model has not seen during training (OOV words). As shown by Hu et al. [12], being able to handle OOV words helps topic models significantly for tasks such as document classification. In WELDA, we can make use of the learned probability distribution for each topic in the embedding space to assign topics to unseen words in the following way: Let's assume we encounter a word previously unseen by our topic model. Instead of ignoring it — as is the default for standard topic modeling — we assign a topic probability based on the learned probability distributions in the embedding space.

2 COMPARING TOPIC MODELS AND WORD EMBEDDINGS

Topic models, as well as word embedding models, exploit the distributional hypothesis [11] which states that similar words tend to occur together. However, both models use this hypothesis to come to different conclusions. LDA works on the bag-of-words assumption, i.e., it treats a document as a whole and ignores the order of the words. When making a prediction for the next word, LDA makes a *global* prediction based on the topic distribution in the current document. Word embedding models, on the other hand, learn a word representation based only on a small, *local* context window around the word.

Both, LDA and word embeddings, can provide a vector representation for words. In LDA, this happens by taking the topic distribution for a word as the embedding in the topic space. However, this representation is not good at keeping linear relationships [19, 20]. Also, it tends to yield sparse vectors as LDA tries to keep the number of topics a word is associated with small.

The different nature of the models is also represented in the standard evaluation method for each. Word embeddings are typically evaluated using word similarity tasks or word analogy reasoning tasks. Topic models, on the contrary, are typically evaluated using topic coherence, i.e., how well the top words from the topics belong together and have a common, easily identifiable theme. When combining topic models and word embeddings, as we propose in this paper, the goal is to incorporate *a priori* semantic information about words into the topic model to increase the coherence of the topics. The assumption is that words which are close to each other in the embedding space should have a higher probability of occurring together in the same topic.

Baroni et al. [3] introduced a classification of count-based methods and prediction-based methods in distributional semantic models. Prediction-based methods try to set word vectors so that they are able to predict the context words. Count-based methods are based on counting the occurrence of words and co-occurrence with other words. Popular count-based methods typically use pointwise mutual information (PMI) and matrix transformation on the co-occurrence matrix. While word embeddings are clearly a prediction-based method, LDA constitutes a hybrid type in this classification. LDA is based on word counts and co-occurrences and treats words as discrete observations, however, the model parameters are chosen to maximize its predictive power.

Another aspect is the interpretability of the model. LDA forces the elements in a vector to sum up to one and all values must be non-negative. Thus, the embedding of a word in the topic space is easily interpretable by humans. With a word vector of [0 0 0.2 0.8] and given the meanings of a topic, a word can be interpreted, for example, as being used 20 % in *sports* and 80 % in *politics*. When working with word embeddings, a vector such as [-2.4 0.3 1.3 -0.1] is not interpretable. The arbitrary dimensions and values of a word embedding vector cannot be directly mapped to meaning by humans.

Regarding the performance, LDA operates much slower than word embeddings when the training set size is equal. LDA becomes expensive on large data sets, partially because the inference problem is NP-hard [25]. For training, the entire corpus and the topic assignments have to fit in main memory for standard implementations. In addition, when the number of documents is increased, the number of topics to represent the corpus usually needs to be increased as well which further slows down the inference. Word embeddings, on the other hand, have been successfully trained on corpora with about 100 billion words [18], thus showing that they scale to large data sets.

3 RELATED WORK

The recent popularity of word embeddings has led to a re-evaluation of topic models. There are two directions for research in this area. One type of models aim to improve word embeddings by incorporating ideas from topic modeling [9, 17, 21, 26]. The other type of models try to improve topic models by incorporating prior knowledge via word embeddings. Since we focus on the latter, we discuss related work in this area in more detail. Topic modeling for short documents, e.g. for classification, can be improved by preprocessing the short texts using word embeddings and then training a topic model on the resulting less sparse space [12]. We aim at a more general improvement of topics and therefore did a comparative evaluation with four recently proposed state-of-the-art topic models that incorporate word embeddings in one way or the other.

Gaussian LDA.. In this approach, a topic is no longer a distribution over words in the vocabulary, but a Gaussian distribution in the word embedding space [10]. Further, words are no longer atomic units but are represented by their corresponding pre-trained word embeddings. Each topic k is associated with a mean vector $\mu_k \in \mathbb{R}^M$ and a covariance matrix $\Sigma_k \in \mathbb{R}^{M*M}$, with M being the word embeddings dimensions. The prior distribution over the means is a normal distribution and the prior distribution over the covariances is an inverse Wishart distribution $\mathcal{W}^{-1}$, a distribution over symmetric positive-definite matrices. Even with some proposed performance improvements, inference in Gaussian LDA is slow and infeasible for high word embedding dimensions.

Latent Feature Topic Models (LFTM).. Nguyen et al. [22] use word embeddings to sample words not only from the multinomial topic distribution but also from the embedding space. Instead of directly sampling a word from the topic-word distribution of the chosen topic, they introduce a Bernoulli parameter $s \sim Ber(\lambda)$ to decide whether the word is sampled as usual from the topic-word distribution or from the latent feature vectors. When sampling in the embedding space, the authors need to define a distribution over all the words. This is achieved by introducing one topic vector τ_t for all topics. This topic vector lies in the same space as the word embeddings. Then, to sample in the embedding space, the model samples from a softmax-normalized multinomial word distribution. The optimization of the topic vector and the coupling between the two components make the inference difficult. As Li et al. [16] state, "the implementation is slow and infeasible when applied to a large corpus".

Nonparametric Spherical Topic Model (NSTM).. Batmanghelich et al. [4] build on top of Gaussian LDA, but argue that Gaussian distributions do not capture the distribution of words in the embedding space well. Instead, they propose the von Mises-Fisher distribution to model topics in the embedding space. The von Mises-Fisher distribution is a distribution on the hypersphere with a mean direction μ and a concentration parameter κ. The distribution is rotationally symmetric around μ. Intuitively, the von Mises-Fisher distribution is a good distribution choice because its distance function is based on cosine similarity. However, semantic relatedness between words is usually not measured by Euclidean distance in embedding models, but rather by cosine distance [19]. The length of the word vectors does not matter for the similarity. In contrast to the previous models, this model is not based on LDA but rather on hierarchical Dirichlet processes. These models allow the number of topics to be determined automatically. The authors propose an efficient parameter estimation algorithm based on stochastic variational inference. They use 50-dimensional word embeddings which they trained on Wikipedia.

Generative Topic Embedding (TopicVec). The main idea compared to LDA is to take the immediate context of a word into account and not only the bag-of-words information [16]. While LDA picks a word from one of the K multinomial topic distributions, TopicVec defines this distribution based on the context words, similar to the word2vec skip-gram architecture. For that, they encode a topic with a vector t_k in the embedding space. The authors provide a variational inference algorithm for TopicVec, which learns the topic assignments and the topic embeddings simultaneously. As the words are sampled from their context words, the architecture allows learning topics for a single document. No word collocation information over different documents is necessary. However, as no other method offers this possibility, we will evaluate TopicVec with one set of topics for all documents.

In this paper, we follow a different approach by building on top of LDA and using the information of the embedding space to resample individual topic words to incrementally improve the topics of the topic model. While other approaches suffer from long runtimes and handling large corpora, we show that combining topic models and word embeddings can be done in an efficient way.

4 WELDA — WORD EMBEDDING LATENT DIRICHLET ALLOCATION

The different definition of context in both models leads to different notions of word similarity: in topic models, words are similar if they have a high probability in the same topic; in word embedding models, words are similar if they are in the vicinity of each other in the embedding space.

4.1 Word Embeddings for Topic Modeling

We aim to combine the two methods to achieve better topic models. As word embeddings and LDA topic modeling are techniques from different backgrounds and with no inherent relationship to each other, we started with exploring how the results of one can

be interpreted in the other model. To this end, we ran initial experiments on the Wikipedia corpus[2], on which we trained both a topic model and a skip-gram word embedding model with two hundred dimensions. We then evaluated the similarities between the words to understand the different strengths of both models. For embedding models, we used the cosine similarity between the two word vectors. For topic models, we built the topic distribution of a word by retrieving the word's probability in all topics in the topic-word matrix. We then normalized this probability to sum to one. To calculate the similarity between two words, we calculated the Jensen–Shannon divergence (JSD) and use $1 - JSD(P \mid\mid Q)$ as the similarity metric.

Even though both, word embeddings and topic models, exploit the distributional hypothesis, the correlation of word similarities is surprisingly low. We calculated the similarities of all word pairs created from the top ten words of an LDA topic model trained on Wikipedia. The similarities of word embeddings and the topic model only correlate with $R_{Pearson} = 0.30$. The results were similar when we used Hellinger distance or Bhattacharyya distance as the similarity measure for probability distributions in the topic model.

When looking at example pairs of high/low similarity in both models in Table 1[3], we see that topic models are able to detect more far-fetched, less obvious connections. This is a good property of topic models, however it sometimes makes the main theme hard to understand. Embedding models, on the other hand, assign high similarity to immediately obvious word pairs, often direct replacements or synonyms of each other.

We further evaluated both model types on a standard similarity benchmark, the WordSim-353 [1] data set. This data set contains 353 word pairs, whose similarity was assessed by thirteen to sixteen human annotators each. The subjects were asked to rate the pair from zero (totally unrelated words) to ten (very much related or identical words). The highest-rated pair was *midday – noon* (9.29) and the lowest-rated pair was *king – cabbage* (0.23). We calculated the correlation between the human ratings and the model's ratings for both the embedding and topic model using the above-mentioned similarity metrics. Embedding models excel at this task, matching the human ratings with $r_{Pearson} = 0.66$, while the topic model only achieves $r_{Pearson} = 0.49$.

To summarize, syntactic variations and more direct connectedness play a bigger role in word embedding models, while topic models often capture more loose relationships. Embedding models are good at assessing similarity between words. These observations make us believe that word embeddings can help create more coherent topic models. Our goal is to strengthen the topic themes by augmenting the top topic words with similar words from the word embedding space, but at the same time not disregarding the bag-of-words collocation information in the documents completely. In this way, we help the user understand the common theme in a topic better while still creating an overview over the entire corpus by looking at all topics.

Table 1: Examples for Different Notion of Word Similarity

		Word Embedding Similarity	
		High	Low
Topic Model Sim.	High	disease, diseases professor, lecturer poetry, prose	games, statistics war, commander rights, discriminated
	Low	catalan, galician prestige, respectability obscenity, sedition	*[completely unrelated words]*

Table 2: Notation

Symbol	Description
K	Number of topics
D	Number of documents
W_d	Number of words in document d
Θ_d	Topic distribution for document $d \sim Dirichlet(\alpha)$
Φ_k	Word distribution for topic $k \sim Dirichlet(\beta)$
Ω_k	Embedded topic distribution for topic $k \sim \mathcal{N}(\mu_k, \Sigma_k)$
$\Psi_{d,i}$	Coin toss $\sim Bernoulli(\lambda)$
$w_{d,i}$	Word in document d at position $i \sim Multinomial(\Phi_{z_{d,i}})$
$w^*_{d,i}$	Word in embedding space to replace $w_{d,i} \sim \mathcal{N}(\mu_k, \Sigma_k)$
$z_{d,i}$	Topic assignment for word $w_{d,i} \sim Multinomial(\Theta_d)$
	Hyperparameters
α, β	Dirichlet priors
λ	Resample probability
μ_k, Σ_k	Mean and variance for topic k in embedding space

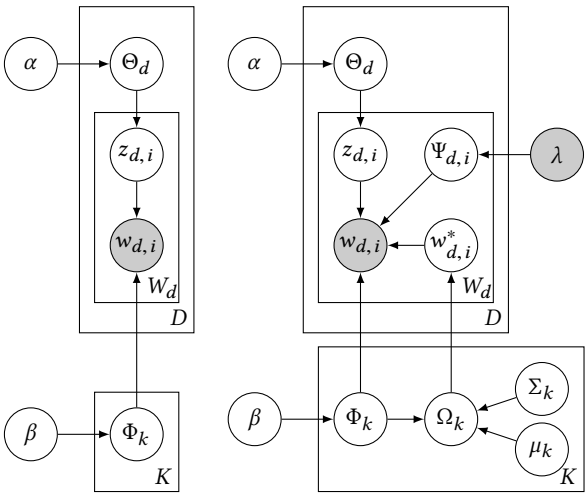
4.2 Generative Model

Compared to standard LDA, WELDA incorporates knowledge from pre-computed word embeddings to estimate the final topic distributions. This ensures that words associated with a topic are also topically coherent in the embedding space. With the used notation, which is shown in Table 2, the generative view of the model is as follows:

(1) Choose number of topics K in the document collection
(2) Choose word distributions $\Phi_k \sim Dirichlet(\beta)$
(3) Fit normal distributions Ω_k to Φ_k for each topic k in the embedding space
(4) Now, for each document d:
    (a) Choose topic distribution $\Theta_d \sim Dirichlet(\alpha)$
    (b) For each position i in the document
        (i) Choose topic $z_{d,i} \sim Multinomial(\Theta_d)$
        (ii) Choose word $w \sim Multinomial(\Phi_{z_{d,i}})$
        (iii) Toss a coin $\Psi_{d,i} \sim Bernoulli(\lambda)$
        (iv) if $\Psi_{d,i} = 1$
            - Replace word w by word $w^* \sim \mathcal{N}(\mu_k, \Sigma_k)$

The crucial step is to replace words in the topic distributions by words that are topically more similar based on the learned probability distributions in the embedding space. This is achieved by the following process: in general, word embedding models are trained

[2] Complete English Wikipedia dump from June 21st, 2016
[3] High similarity indicates a similarity value in the upper quartile; low similarity in the lower quartile.



(a) LDA **(b) WELDA**

Figure 2: Plate notation for LDA and WELDA

on predicting words at an exact position. That means, if we look at similar words for a given word, the model predicts a word which could have been at the same position. We want to exploit this knowledge about similar words encoded in the word embedding space by transferring probability mass from general words in the topic model towards topically similar words from the embedding space. We achieve this by sampling new words from the embedding space. For this, we learn a multivariate continuous distribution in the embedding space for each topic. We use Gaussian distributions for that, but other distributions can be used as well.[4] We call each of these distributions Ω_k an *embedded topic distribution*. This embedded topic distribution is estimated from the top words of each topic Φ_k and their representation in the embedding space. While iterating over the document collection during training of the model, we selectively replace words $w_{d,i}$ in the documents with words $w^*_{d,i}$ from the embedded topic distribution. We achieve this by sampling a new word in the innermost loop of the Gibbs sampling algorithm and then proceed as if that newly sampled word appeared. These words have a high similarity in the embedding space to the top words in the topic and thus are excellent candidates to replace overly general words within the topic model. Before running Gibbs sampling, we load a pre-trained word embedding model, e.g. one learned on the English Wikipedia, but any other pre-trained embedding model would do[5].

4.3 Inference

We use the standard LDA sampling equation for our collapsed Gibbs sampler:

$$p(z_{d,i} = k | z_{-(d,i)}, \mathbf{w}) \propto \frac{\mathbf{n}^{(\mathbf{w_i})}_{-\mathbf{i,j}} + \beta}{\mathbf{n}^{(\cdot)}_{-\mathbf{i,j}} + \mathbf{W}\beta} \frac{\mathbf{n}^{(\mathbf{d_i})}_{-\mathbf{i,j}} + \alpha}{\mathbf{n}^{(\mathbf{d_i})}_{-\mathbf{i},\cdot} + \mathbf{K}\alpha} \tag{1}$$

[4]We also experimented with multivariate Gaussian mixture distributions and multivariate von Mises-Fisher distributions; see Section 5.2

[5]We also investigated the use of word embedding models trained on the topic model training data; see Section 5.3

where $\mathbf{w}$ are all words in the corpus, $n^{(w_i)}_{-i,j}$ is the number of times w_i has been assigned to topic j, and $n^{(d_i)}_{-i,j}$ is the number of times a word from document d_i has been assigned to topic j. After initializing our model by running the standard LDA algorithm until convergence to obtain stable topic assignments, we estimate Gaussian distributions in the embedding space based on the top N_{top} words of each topic. Then we run additional Gibbs sampling iterations using our collapsed Gibbs sampler but exchanging some of the words in the document based on a Bernoulli distribution. If this coin flip with success probability λ succeeds, we sample a vector from the word embedding space, based on the topic distribution of the new topic of the current word. A nearest neighbour search in the embedding space yields the nearest word w^*. Subsequently, we update the counts in the document-topic matrix and in the topic-word matrix, as if we had observed the sampled word w^* and not the word w that we actually observed in the document.

4.4 Out-of-Vocabulary Words

One of the big advantages of WELDA is its ability to cope with unseen words. In contrast to standard LDA, where words that were not observed during training are simply ignored, WELDA can handle unseen words and assign topic probabilities to them. During an inference step to assign topics to words of a new document, unseen words during training are assigned a membership probability for each topic according to the learned Gaussian distribution. Then a particular topic can be sampled for the new word using Gibbs sampling. Figure 3 (center and right) shows an example from the 20News corpus. We can identify two topics: "shipping&handling" (green) and "computers" (blue) together with their computed Gaussian distribution indicated by grey circles. If we now want to infer the topic distribution of a new document and observe a previously unseen word, e.g. "buy", we can locate this word in the embedding space and assign topic probabilities to this word. In our example, "buy" would get a higher probability in the blue "computer"-topic than in the green "shipping&handling"-topic and nearly no probability in the red topic.

4.5 Reducing Computational Complexity

In order to ensure fast learning of the model parameters we propose some simplifications. For example, we limit the number of topic words N_{top} that we consider for the estimation of the Gaussian distributions in the embedding space. This can be done without much harm, since the top 100 to 200 words of a topic are usually most indicative. Further, we use principal component analysis (PCA) to project the word vectors to a lower number of dimensions N_{pca} to speed-up the necessary nearest neighbour search. Finding the nearest neighbour becomes necessary after we obtained a sample from the embedded topic distribution and we want to find the closest word to our sample in the embedding space. The naive algorithm uses linear search and evaluates all words in the embedding space. This is expensive, especially because it happens in the innermost loop of Gibbs sampling and is therefore executed very often, approximately $\lambda \times N_{iter} \times N_{words}$ times, where N_{words} is the number of words in the entire training corpus, N_{iter} the number of Gibbs sampling iterations and λ the resampling probability. However, there are methods with a better runtime: we use either k-d trees

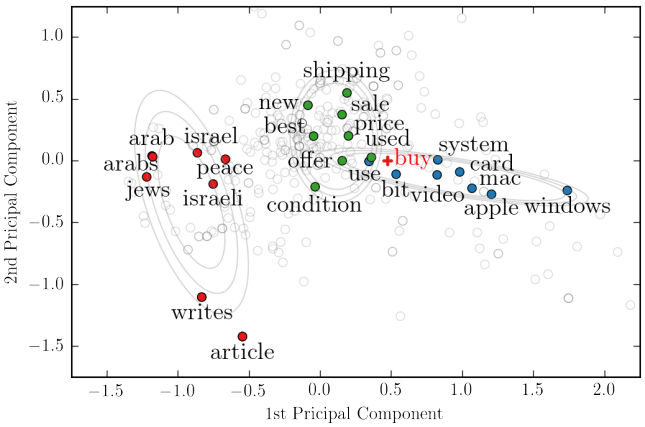


Figure 3: Illustration of three LDA topics in the word embedding space together with the grey isolines of the learned probability distributions. On the left you see two stop words assigned to the red topic. On the right you see two topics close together in the embedding space and a new word "buy", which was not in the training data.

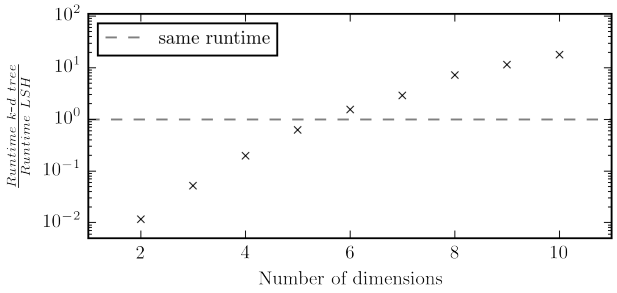


Figure 4: Performance comparison of LSH and k-d tree for the nearest neighbour word search based on the number of PCA-dimensions N_{pca}

or locality-sensitive hashing depending on the number of dimensions k of the embedding space. k-d trees by Bentley [6] arrange the points in a tree structure which allows retrieving the nearest neighbour in $O(\log k)$ on average. k-d trees are guaranteed to find the nearest neighbour. However, for high dimensions, the search degenerates to the linear search due to the curse of dimensionality. Thus we use locality-sensitive hashing (LSH) by Indyk et al. [13] for higher dimensions. LSH is an approximate method, i.e., it is not guaranteed that the nearest neighbour is found. LSH uses a hash-function based on random hyperplanes in the space to bucketize all vectors. Then, for a new vector, the same hash function is applied and only the bucket with the corresponding hash value must be checked. LSH has the opposite property, i.e., the lower the number of dimensions the longer it takes.

We therefore evaluated the runtime of the two approaches during a WELDA run. Figure 4 shows the ratio of the runtime of the k-d tree and LSH. Note the logarithmic y-axis, so the ratio actually increases almost exponentially. We see that up to five dimensions

Table 3: Top Words of Background Topic for 20News Corpus

would one writes like article think get know time people even good much well way make see really want apr right also going still say sure since first something back thing never take many better may things

including the k-d tree is faster for looking up the nearest neighbour. For six or more dimensions, LSH is preferred. We therefore switch the algorithm depending on the number of pca-dimensions N_{pca} we use in our experiments.

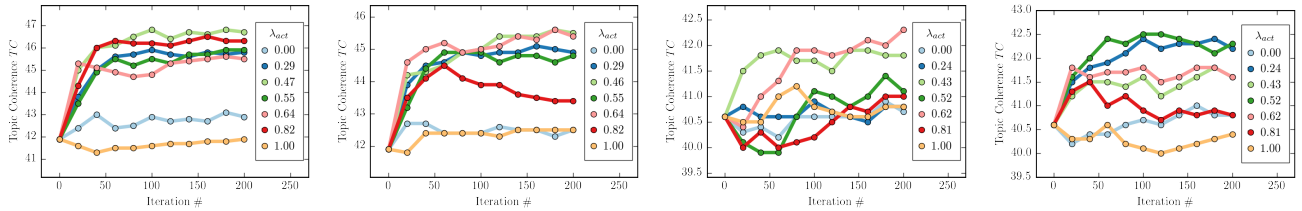
4.6 Background Topic for Corpus-Specific Stop Words

To speed up the topic improvement process, we include a corpus-specific stop word topic to capture not only more general English-language stop words but also corpus-specific ones. We detect these stop words automatically during the initial LDA Gibbs sampling. Instead of using a symmetric Dirichlet α prior for all topics, we multiply the α_0 prior by a factor α_0^{boost}. By increasing the value of α_0 we force the first topic to appear in almost all documents. This forces LDA to put words in this topic, which occur in many documents and are rather generic. We call this first topic the *background topic* because it occurs in many documents as the generic word background for the more informative topics.

We experimented with different values of α_0^{boost}. Good results were obtained for $\alpha_0^{boost} = 15$ which led to the background topic occurring in more than 90% of all documents. Indeed, if we look at the top words from the background topic in Table 3, we find many corpus-specific stop words, such as "article" and "people".

In addition, the top-50 words in the background topic get a special treatment. Whenever one of these words occurs, we always resample the word, even if the coin toss in the WELDA algorithm failed. Note that we keep this list fixed during the WELDA iterations. The background topic changes quickly due to this approach and a robust topic emerges after a few iterations. Due to this adapted sampling technique, we differentiate between λ and λ_{act} in experiments. The former is the λ the algorithm was configured with. The latter represents the actual percentage of replacements when we also replace corpus-specific stop words. Therefore λ_{act} is always higher than λ.

This procedure helps replacing generic words from all topics, thus making the topics better. We hypothesize that WELDA has this property automatically, even without dedicated background topic replacement. This is because, by the nature of sampling, these words are more likely to be replaced as they occur more often in the documents. Also, these words are usually far away from the means of the embedded topic distributions. Thus, these words disappear from the topics during our modified Gibbs sampling. Figure 3 shows original LDA topics in the embedding space. The red topic on the left clearly contains two outliers: corpus-specific stop words ("writes" and "article"). In WELDA these words would get replaced eventually during sampling yielding more robust and stable topics. Using a dedicated background topic and always resample its words accelerates this process, since WELDA is not initialized with LDA



(a) With 20News embeddings (b) With Wikipedia embeddings (c) With Nips embeddings (d) With Wikipedia embeddings

Figure 5: Topic coherence for the first 200 iterations for the 20News-corpus (left) and Nips-corpus (right)

Table 4: Corpora Used for Evaluation

	20News	Nips
Documents	11,295	1,740
Words	1,395,973	2,497,017
Classes	20	*no classes*
Vocabulary Size	51,951	35,171
Words per Document	124	1,435

topics that are cluttered with stop words. In Section 5.2 we show experimental results for the usage of a background topic.

5 EVALUATION

We evaluated and compared our model on two different datasets: 20News and Nips. While these datasets are rather small, most current approaches combining word embeddings and topic models have already problems processing them. To test and compare the various models, we used two different tasks: *topic coherence* and *word intrusion*.

5.1 Datasets

Some statistics of the 20News[6] and Nips[7] datasets are listed in Table 4. 20News is a collection of newsgroup posts from the early 1990s. The data was downloaded from twenty different newsgroups and the name of the newsgroup is taken as the class label. The Nips corpus is a collection of scientific papers from the proceedings of the Neural Information Processing Systems conference. The papers focus on neural computation, learning theory, algorithms and architectures, cognitive science, information theory, neuroscience, vision, speech, control and diverse applications.

We chose these two data sets because they represent two different type of corpora. The 20News corpus is a general corpus with a wide range of topics. The Nips corpus has a narrow focus with many technical terms and a lot of topical overlap. We preprocessed the corpora conducting tokenization, lowercasing, and stop word removal using a list of 150 general words from Python's nltk.

5.2 Parameter Settings

There are several configurable parameters in the algorithm that we will detail in the following.

[6]http://qwone.com/~jason/20Newsgroups/
[7]http://cs.nyu.edu/~roweis/data.html

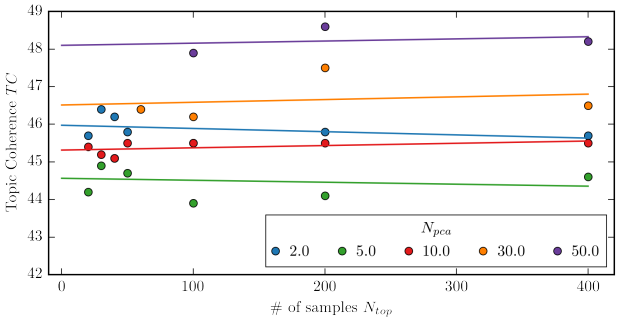


Figure 6: Influence of number of words to consider for distribution estimation N_{top} and number of dimensions for dimensionality reduction N_{pca} on topic coherence

Choice of embedded topic distribution: We tested different distributions: the performance of the *multivariate Gaussian distribution* and the *multivariate von Mises-Fisher distribution* were similar. The *Gaussian mixture model* performed well only if the initial number of topics was chosen too small, i.e. the topics generated by the original topic model were mixtures of multiple topics. With a sufficient number of topics, using mixtures of Gaussians was not necessary. In princliple, WELDA can be used with any kind of multivariate distribution. Since using multivariate Gaussians allows faster training of the model with equal or exceeding quality of resulting topics, we only report results using Gaussians here.

Resampling probability $\boldsymbol{\lambda}$: The parameter λ decides, how often a word is resampled from the topic distributions. Thus, the parameter λ governs how much extra information from the word embeddings is leaked into the topic model. Running WELDA with $\lambda = 0.0$ corresponds to standard, unmodified LDA and will not have any effect on the model because we already start with a converged topic model. With $\lambda = 1.0$, no single word is taken from the documents, rather all words are sampled from the embedding space. Experiments showed that less then 100 iterations were necessary for the model to converge. Figure 5 displays topic coherence for the two datasets with embeddings trained on different corpora. Best topic coherence scores where achieved for $0.4 \leq \lambda \leq 0.6$. Currently we do not optimize this hyperparamter.

Number of top words for distribution estimation N_{top}: We are using the top words of each topic to estimate the embedded topic

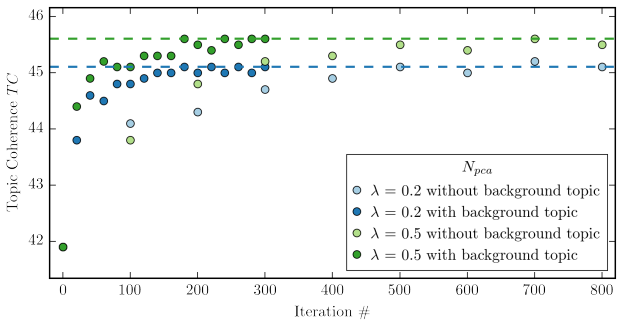


Figure 7: Comparison of WELDA with and without background topic. Background topic resampling converges faster, but not better.

distributions. Usually only the first words of a topic share a common theme.

Number of dimensions for dimensionality reduction N_{pca}: Before estimating the multivariate distribution in the embedding space, we project the high-dimensional embedding space to a lower space. There is a trade-off between the N_{pca} parameter and the N_{top} parameter. The lower the final dimension number, the fewer words we need for estimating the distribution. Fewer words are better because in this case we can use the really good top words at the start of a topic, and not the less probable ones further down the list. On the other hand, by projecting the space to a lower dimension, we throw away semantic information in the embedding space and the similarity between words becomes less reliable. In general, we expect a higher dimensionality to perform better. Best topic coherence was achieved for $N_{pca} = 50$ and $N_{top} = 200$. Figure 6 shows the influence of N_{top} and N_{pca}. Note that the regression lines added to the plot indicate that N_{top} has nearly no influence on the topic coherence measure.

Background topic: We include a dedicated background topic to capture corpus-specific stop words. This has no influence on the topic quality but on the runtime until convergence. Figure 7 shows the convergence behavior on the 20News corpus with Wikipedia embeddings. While WELDA converges with a dedicated background topic after 200 iterations, at least 500 iterations are needed without.

Standard LDA parameters: As in standard LDA, the Dirichlet hyperparameters must be chosen. We use the same parameters in WELDA that we also use for the initial LDA iterations ($\alpha = \beta = 0.02$). We run LDA for 1,500 iterations [22], to ensure that the log-likelihood converges.

5.3 Topic Coherence

Topic coherence is a standard measure to evaluate topic models and measures how well the top-words in a topic fit to each other. We use the C_V metric from Röder et al. [23] which had the highest correlation with human annotators ($r_{Pearson} = 0.731$) in their experiments. We fixed the number of top words to consider for evaluation to $N = 10$ which is a common choice in the literature [2, 23]. All experiments were run with $K = 50$ topics. We tested on the

Table 5: Topic Coherence on the 20News and Nips Corpus with Wikipedia and Corpus-Specific Embeddings

Corpus:	20News		Nips	
Embeddings:	Wikipedia	20News	Wikipedia	Nips
LDA	—— 41.9 ——		—— 40.6 ——	
Gauss. LDA	38.5	40.3	39.2	39.5
LFTM	43.9	45.6	41.6	42.8
NSTM	—	—	37.5	36.8
TopicVec	48.1	46.2	43.4	**43.6**
WELDA	**48.6**	**47.4**	**44.2**	42.9

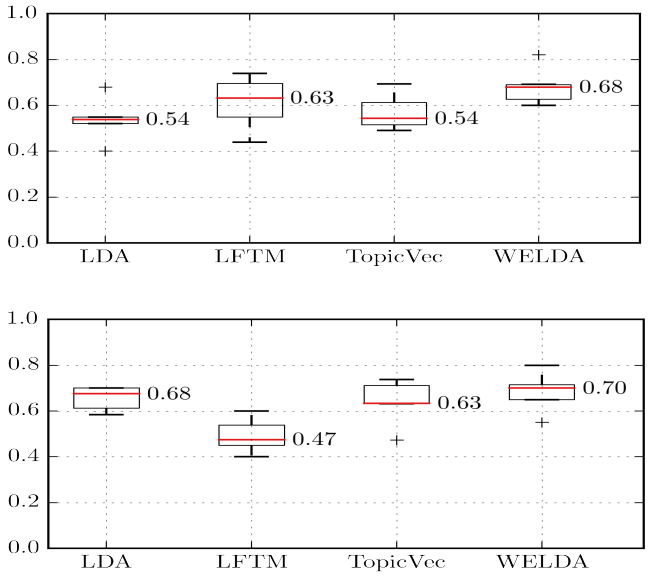


Figure 8: Word intrusion study results for model with 50 topics and 20 topics

20News and on the Nips corpus. We optimized the parameters for all models on both corpora. We run WELDA for 200 iterations with Gaussian distributions and $\lambda = 0.4$. An overview of the results can be seen in Table 5. Using word embeddings learned on Wikipedia yields better results than using embeddings learned on 20News or Nips. One explanation for that is the size of the corpora, with Wikipedia having millions of documents instead of thousands. Due to the nature of the topic coherence measure (which uses Wikipedia word co-occurrence) it is also no surprise that results for the 20News corpus are better. The Nips corpus is too domain-specific to be captured by many Wikipedia articles. For NSTM, the 20News corpus was to large and no results could be obtained. On both corpora (using Wikipedia embeddings), WELDA outperforms the other approaches as well as vanilla LDA significantly.

5.4 Word Intrusion

So far we evaluated the quality of our topics using the topic coherence measure [23] to assess model quality. While the latter is cheap and many models can be evaluated, the correlation with human ratings is not perfect ($r_{Pearson} = 0.731$). Also, it is based on

Table 6: Topic Evolution from LDA to WELDA after 200 Iterations (Crossed-Out Words were Replaced by Bold Ones)

20News
water power ~~use~~ air high ~~system~~ light ~~used heat time~~ **nuclear cooling temperature cycle plants**
year players baseball ~~article good writes~~ lopez ~~jewish~~ league average **hit braves season hitter**
information list mail ~~may~~ internet send anonymous ~~faq~~ email ~~use~~ **system available privacy**
israel israeli jews arab arabs ~~writes article~~ peace ~~would~~ lebanese **lebanon land israelis**
team hockey season nhl players games ~~league~~ game teams ~~year~~ **cup play**
Nips
bayesian gaussian distribution prior posterior ~~data using~~ evidence ~~mean~~ sampling **monte carlos covariance**
kernel ~~algorithm~~ vector support loss margin ~~set~~ function ~~linear~~ examples **gerneralization svm problem**
speech recognition speaker gamma acoustic phoneme ~~time~~ vowel ~~information~~ speakers **generalization segmentation**
spike firing neurons time information neuron ~~rate~~ spikes stimulus ~~neural~~ **spiking trains**
state learning reinforcement policy action value optimal states actions ~~time~~ **reward**

co-occurrence statistics from the English Wikipedia under the assumption that this corpus contains a large number of topics and can therefore be used as a reference corpus. In the case of the 20News corpus with its wide range of general topics, this assumption holds indeed. However, it is questionable for the Nips corpus which covers a very technical, focused domain. For this reasons, we decided to conduct a manual study on the Nips corpus in addition. We evaluated the quality of topics by the word intrusion task defined by Chang et al. [8]. In this task, a subject is presented a list of six words. Five of these six words belong to the top words of one topic of the topic model. The sixth word, called the *intruder*, was chosen randomly from another topic. The idea of this word intrusion study is the following: if the five words form a clear and coherent topic it should be simple to detect the intruder.

In our study, we picked the four best models based on topic coherence, i.e., LDA, LFTM, TopicVec and WELDA (Gaussian LDA and NSTM didn't make the cut). We tested two different scenarios: one with $K = 50$ topics and one with $K = 20$ topics. Our subject group consisted of computer science students and computer scientists familiar with the topics of the Nips corpus. To avoid biased results, subjects were not presented with the information, whether their selection was correct [8]. Further, the underlying topic model was not displayed and we randomized the order of the samples.

We present the results for models with $K = 50$ topics and $K = 20$ topics in Figure 8. For the plots, we calculated the accuracy of each person on each topic model. Each model was evaluated by at least five persons (seven at most), resulting in at least five accuracy values. For both, $K = 20$ and $K = 50$, WELDA achieves the highest accuracy (around 70%), outperforming the other approaches and LDA significantly.

5.5 Topic Quality

Besides the quantitative evaluation results in the previous sections, we also want to report qualitative results to demonstrate the functioning of our model. Table 6 presents the top ten words of three random topics of the 20News (top) and Nips (bottom) corpora. The examples clearly show WELDA's capability to add specific, meaningful words among the top words, while at the same time downgrading overly general words.

Table 7: Runtime comparisson: 20News+Nips

Method	Runtime
Gaussian LDA	4283 min
LFTM, $\lambda = 0.6$	3753 min
NSTM	DNF
TopicVec	294 min
WELDA, $\lambda = 0.5$	206 min

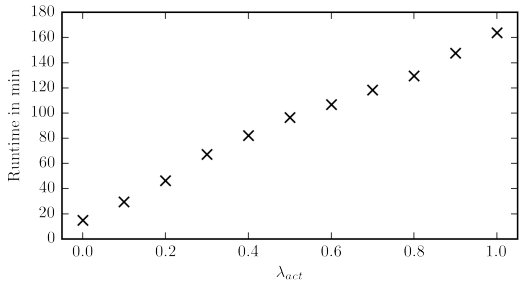


Figure 9: Runtime of WELDA on the 20News corpus depending on the resampling Parameter λ with Wikipedia embeddings and $N_{pca} = 10$

5.6 Runtime

Our Experiments were conducted on a 32-core, Intel Xeon CPU E7-8837 @ 2.67GHz machine with 256 GB main memory. While some techniques have parallelized implementations available, others have not. Therefore, to ensure comparability, we ran all approaches with only a single core. We tested on 20News with 200-dimensional embeddings (50-dimensional for Gaussian LDA) and on Nips with 50-dimensional corpus-specific embeddings, both with $K = 50$ topics. We present the runtimes of the different approaches in Table 7. On both corpora, WELDA is the fastest, in total 30% faster than the second fastest approach (TopicVec) and an order of magnitude faster than Gaussian LDA and LFTM. LFTM and Gaussian LDA are very slow with 200-dimensional embeddings, making them infeasible to run on larger corpora, while NSTM did not finish at all since the 20News corpus was too large to process for NSTM.

Note that WELDA's performance largely depends on the resampling probability λ. The higher λ the more new words are sampled which leads to more nearest-neighbour searches which dominate the runtime (see Figure 9). Still, even with $\lambda = 1.0$ the runtime of WELDA on 20News is with 164 min still faster than the next-fastest: TopicVec (178 min).

6 CONCLUSIONS

In this paper, we have shown that word embeddings can be used to improve topic models. We started from the distributional hypothesis and analyzed its different realization in topic models and in word embedding models. Based on this analysis, we proposed WELDA, a new type of topic model aiming to utilize the strengths of both model families. WELDA learns topic distributions for all topics in the word embedding space. These topic distributions capture and generalize the positions of the words in the embedding space. By learning an entire distribution and not only a single topic vector, we are also capturing the covariance of the word positions in the embedding space. We exchange document words by resampling words from this distribution during Gibbs sampling inference. We shift probability mass from general, low-informative words to specific, more salient words that strengthen the main theme of each topic.

Our evaluation showed improved results over state-of-the-art combined topic models on the task of detecting word intrusion and with respect to topic coherence. We could further confirm the assumption that higher dimensionality of the word embeddings leads to better performance, as more semantic regularities can be encoded in the embedding space. We experimented with using corpus-specific embeddings instead of pre-trained embeddings from Wikipedia which gave slightly worse results. Albeit lacking quality when evaluated on their own, the corpus-specific embeddings can still help to improve topic coherence, because they are learned on the same domain as the topic model. While other state-of-the-art combinations of word embeddings and topic modeling (except TopicVec) can only handle very small corpora in a reasonable amount of time, our approach is by far the fastest.

REFERENCES

[1] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pas, and Aitor Soroa. 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In *Proceedings of the Conference of the North American Chapter of the ACL (NAACL)*. ACL, 19–27.

[2] Nikolaos Aletras and Mark Stevenson. 2013. Evaluating Topic Coherence Using Distributional Semantics. In *Proceedings of Conference on Computational Semantics (IWCS)*. ACL, 13–22.

[3] Marco Baroni, Georgiana Dinu, and German Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*. ACL, 238–247.

[4] Kayhan Batmanghelich, Ardavan Saeedi, Karthik Narasimhan, and Sam Gershman. 2016. Nonparametric Spherical Topic Modeling with Word Embeddings. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*. ACL.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research* 3 (2003), 1137–1155.

[6] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (1975), 509–517.

[7] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *The Journal of Machine Learning Research* 3 (2003), 993–1022.

[8] Jonathan Chang, Sean Gerrish, Chong Wang, J. L. Boyd-graber, and David M Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. In *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., 288–296.

[9] Jianpeng Cheng, Zhongyuan Wang, Ji-Rong Wen, Jun Yan, and Zheng Chen. 2015. Contextual Text Understanding in Distributional Semantic Space. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*. ACM, 133–142.

[10] Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for Topic Models with Word Embeddings. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*. ACL, 795–804.

[11] John Rupert Firth. 1957. *Papers in linguistics, 1934-1951*. Oxford University Press.

[12] Weihua Hu and Jun'ichi Tsujii. 2016. A Latent Concept Topic Model for Robust Topic Inference Using Word Embeddings. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*. ACL, 380–386.

[13] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors : Towards Removing the Curse of Dimensionality. In *Proceedings of the Symposium on Theory of Computing (STOC)*. ACM, 604–613.

[14] Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*. ACL, 302–308.

[15] Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*. ACL, 171–180.

[16] Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. 2016. Generative Topic Embedding: a Continuous Representation of Documents. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*. ACL, 666–675.

[17] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical Word Embeddings. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2418–2424.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., 3111–3119.

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

[20] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Vol. 13. ACL, 746–751.

[21] Christopher E. Moody. 2016. Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec. *CoRR* abs/1605.02019 (2016).

[22] Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics (TACL)* 3 (2015), 299–313.

[23] Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the Space of Topic Coherence Measures. In *Proceedings of the Conference on Web Search and Data Mining (WSDM)*. ACM, 399–408.

[24] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323 (Oct. 1986), 533–536.

[25] David Sontag and Daniel Roy. 2011. Complexity of Inference in Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., 1008–1016.

[26] Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2015. Learning Word Representations by Jointly Modeling Syntagmatic and Paradigmatic Relations. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*. ACL, 136–145.