Masterarbeit

English Title

German Title

Stefan Bunk

stefan.bunk@student.hpi.uni-potsdam.de

Eingereicht am 23.12.2016

Fachgebiet Informationssysteme
Betreuung: Dr. Ralf Krestel

Abstract

Twitter, a popular micro-blogging service, allows sharing news quickly and by nearly everybody. Users often express their opinion on current news and report on events they participated in. Tweets belonging to a news topic provide opportunities for journalists, social scientists, and common users. Furthermore, the analysis of these tweets enables new business applications. In this research, we investigate how tweets related to a specific news event can be identified and used for providing benefit to a user. We pursue two main goals: the search of explicitly or implicitly related tweets for a given news article and their presentation and aggregation.

For the identification of tweets, we develop a family of search strategies that produce keyword-based search queries applicable to the Twitter Search API. For the estimation of the search term and query quality, the strategies make use of a corpus of current articles and corresponding, explicitly linking tweets. To measure the quality of the developed approaches, we perform an automatic evaluation using an approximatively gold-standard. Additionally, we carried out a user study with more than 30 participants and 1,300 ranking submissions. The evaluation results show that the strategies perform well in terms of relevance and interestingness of found tweets.

Simplifying the handling of the search results, we implemented a tweet presentation and diversification strategy providing the user with an overview of interesting tweets. Additionally, we explore how the knowledge hidden in tweets can be uncovered and represented. For this purpose, we develop strategies that summarize tweets based on topical, spatial, temporal, and sentiment features. Visualizations implemented on top of these results allow gaining new insights about the underlying news story, like shifts in public mood or the development of public interest.

The presented strategies are implemented in a prototypical web application called "TweNew". The application provides access to the developed search and recommendation strategies as well as to the aggregation and visualization techniques. Additionally, we discuss the extension of the prototype to a real-time search engine.

Zusammenfassung

Twitter, ein beliebter Mikroblogging-Dienst, ermöglicht es Nachrichten schnell und durch jedermann zu verbreiten. Nutzer berichten dabei häufig über Ereignisse an denen sie teilnahmen oder drücken ihre Meinung über aktuelle Geschehnisse aus. Tweets mit Bezug zu Nachrichtenthemen stellen eine wichtige Wissensquelle für Journalisten, Soziologen und gewöhnliche Nutzer dar. Durch die Analyse dieser Tweets werden außerdem zahlreiche Geschäftsanwendungen möglich. In der vorliegenden Arbeit erforschen wir wie nachrichtenbezogene Tweets identifiziert werden können und wie Nutzer davon profitieren können. Wir verfolgen dabei zwei Ziele: die Suche aller explizit oder implizit nachrichtenbezogenen Tweets für einen gegebenen Zeitungsartikel sowie die Empfehlung und Aggregation der Suchergebnisse.

Wir entwickeln dazu eine Familie von Suchstrategien, die, basierend auf Schlüsselworten, Suchanfragen für die Twitter Search API generieren können. Zur Abschätzung der Schlüsselwort- und Anfragequalität nutzen die Strategien einen Korpus von Artikeln und damit explizit verlinkten Tweets. Zur Evaluierung der vorgestellten Ansätze nutzen wir einen approximativen Goldstandard. Außerdem wurde eine Nutzerstudie mit mehr als 30 Teilnehmern und 1.300 eingegangenen Bewertungen durchgeführt. Die Ergebnisse belegen, dass die untersuchten Ansätze Tweets finden, die sowohl als relevant als auch als interessant empfunden werden.

Um dem Nutzer eine Übersicht über die gefundenen Tweets bereitzustellen, entwickeln wir eine Empfehlungs- und Diversifizierungsstrategie. Zusätzlich erforschen wir, wie verwertbare Informationen aus den Suchergebnissen gewonnen und dargestellt werden können. Zu diesem Zweck entwickeln wir Strategien, die Tweets anhand von zeitlichen, räumlichen, thematisch- und stimmungsbasierenden Merkmalen zusammenfassen. Auf deren Ergebnissen basierende Visualisierungen ermöglichen es Nutzern neue Einsichten über das behandelte Thema zu bekommen. So können zum Beispiel die Veränderungen der öffentlichen Meinung erkannt oder die Entwicklungen des öffentlichen Interesses nachvollzogen werden.

Die vorgestellten Verfahren werden als Prototypen in einer Webapplikation namens “TweNew” umgesetzt. Die Anwendung ermöglicht den Zugriff auf die entwickelten Such- und Empfehlungsmethoden sowie Aggregationen und Visualisierungen. Außerdem erörtern wir eine Erweiterung der Applikation zu einer echtzeitfähigen Suchmaschine.

Contents

1 Introduction

A core task in natural language processing (NLP) is to understand the meaning of words. Many downstream NLP tasks benefit from this, for example, text categorization, part-of-speech tagging, and machine translation. A popular concept to qualify the meaning of a word is to look at the contexts in which the word appears. A famous quote by Firth says: “You shall know a word by the company it keeps” [**Firth1957**]. This assumption is also known as the distributional hypothesis.

Two existing approaches in this area are topic modeling and word embeddings. Topic modeling assumes that an author has certain topics in mind when writing a document, which are chosen beforehand. For example, an author could decide to write a document about *politics* (70 %) in *sports* (30 %) and then picks the words “coalition”, “election” and “corruption” for *politics* and the words “soccer” and “ball” for *sports*. Topic modeling techniques, with latent Dirichlet allocation being the most popular one, try to recover the hidden topics in large corpora, i.e. find out that “soccer” and “ball” come from the same topic. As topic models are probabilistic models and hence provide probability distributions as the result, the topics can be easily interpreted by humans.

With word embeddings each word is assigned a vector in a high-dimensional vector space. These vectors are automatically learned using neural networks. During the supervised training each word must try to predict its surrounding words. This way, the context of a word is encoded in the vector representation of a word. Using a clever network architecture, these vectors exhibit interesting properties. First, similar words tend to be in similar positions in the vector space. For example, when looking for the most similar words for “France” using cosine distance, the model outputs “Spain”, “Belgium”, “Netherlands” and “Italy”. Second, the word vectors exhibit interesting linear relationships. For example, when calculating the vector $\mathit{vector}("King") - \mathit{vector}("Man") + \mathit{vector}("Woman")$ and looking for the closest word at the vector result, the result is the word “queen” [**Mikolov2013b**].

The two approaches have different origins. Word embeddings have their roots in the neural network and deep learning community, while topic modeling stems and Bayesian statistics. Hence, there is relatively little research in combination of these two methods. In this master thesis, we aim to explore potential synergies between these technologies. In particular, it might be useful to investigate “whether the two types of models are complementary in the errors they make, in which case combined models could be an interesting avenue for future work” [**Baroni2014**].

This master thesis is organized as follows: Chapter **??** will introduce topic modeling and word embeddings, compare the two approaches and show existing work in combining the two methods. Chapter **??** will present first experiments, which we want to conduct at the start of this master thesis. Chapter **??** will show our approach for this master thesis.

2 Related work

2.1 Topic modeling

Many methods for modeling natural language text have been proposed. An early popular method was the tf-idf [**SparckJones1972**] scheme, which reduced each document in a corpus to a vector of term frequencies normalized by inverse document frequencies. Later, latent semantic analysis (LSA) [**Deerwester1990**] was proposed to reduce the large matrix to a smaller subspace using singular value decomposition. While these approaches have their origins in linear algebra, topic models have been developed from a probabilistic view to allow better interpretation. Topic models are generative probabilistic models of a document collection, which assume hidden topics have guided the generation of the text. In this way of thinking, an author picks certain topics to write about, and then picks words associated with these topics for a specific document. The goal is then to uncover the unseen, so called "latent", topics from a text corpus. All of these approaches have in common that they assume a bag-of-words representation of documents.

2.1.1 Earlier approaches

A basic generative topic model is the mixture of unigrams model [**Nigam2000**]. In this model, a topic is a fixed probability distribution over the vocabulary. However, only one topic is allowed per document, which limits its predictive power.

A more sophisticated topic model is probabilistic latent semantic analysis (pLSA), which allows multiple topics per document. The model is fitted with an expectation-maximization algorithm. However, pLSA does not provide a full generative model applicable to unseen documents. Also, as the number of parameters grows linearly with the number of training documents, it tends to overfit [**Blei2003**].

2.1.2 Latent Dirichlet allocation (LDA)

LDA was proposed by Blei et al. [**Blei2003**] for modeling text corpora and other collections of discrete data. In an unsupervised fashion, it can automatically detect similarities between words and group them in one topic. The number K of topics is constant and must be chosen beforehand.

LDA defines a topic as a distribution over a fixed vocabulary. Different topics assign different probabilities to the same word, for example, the word "soccer" would have a much higher probability in a *sports* topic than in a *politics* topics. The opposite would hold for the word “coalition”. Note that a topic is only a distribution over words, i.e. LDA does not generate a name or a summary for a topic.

A document is assumed to consist of one or several topics with a certain, fixed distribution. A document could be 20 % about *sports* and 80 % percents about *politics*, or 30 % about *climate*, 25 % about *cars* and 45 % about *politics*. In general, LDA aims to generate sparse distributions for both cases, i.e. a topic should have only a few words (relative to the vocabulary of the entire corpus) with high probability and a document should have only a few topics. This is governed by two scalar hyperparameters α and β, which influence two symmetrical prior Dirichlet distributions.

With these two hyperparameters, the generative document creation process of LDA can be summarized as follows:

1. Fix number of topics K in the document collection
2. Choose word distributions $\phi_j \sim Dirichlet(\beta)$ for each topic $j \in \{1 .. K\}$
3. 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 $w \sim Multinomial(\phi_{z_{d,i}})$ for current word

This process is also illustrated in the probabilistic graphical model in plate notation in Figure 2.1. This model can also be seen as a soft-classification of documents: instead of assuming exactly one topic per document, each word in a document can potentially come from a different topic.

In reality, we only observe the words in the documents, and need to reverse engineer the hidden, “latent” parameters of the model, namely the word distribution per topic ϕ, the topic distribution per document θ and the word-topic assignments z. In this process, called *inference*, we try to find the most probable parameter assignments, which could have generated our document collection. Computationally, this is equivalent to finding the posterior distribution of the parameters given the observed words. Blei et al. [**Blei2003**] propose an algorithm based on variational approximation, however Griffiths and Steyvers [**Griffiths2004**] proposed an algorithm based on Gibbs sampling.

The output of running LDA on a corpus is three-fold:

- Topics as set of similar words, e.g. the terms “soccer”, “ball”, “league” could form a topic named *sports*.
- Topic distribution of a single word, e.g. the term “jaguar” occurs 80 % in the *cars* topic and 20 % in the *animals* topic
- Topic distribution inside a specific document or inside the entire corpus

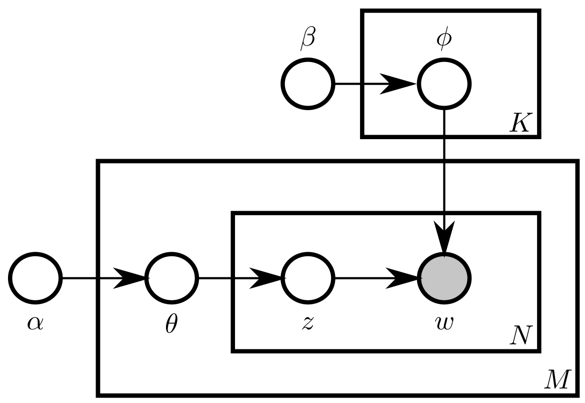

Figure 2.1: Graphical model for latent Dirichlet allocation with K topics, M documents and N words in a document.

LDA has been applied to many domains, namely document modeling, text classification, collaborative filtering [**Blei2003**] and document summarization [**Wang2009**]. Other use cases include population genetics [**Pritchard2000**] and computer vision [**LiFei-Fei2005**].

Implementations There exist implementations of LDA in Java, C/C++, Python and Matlab. In this thesis, we will use two particularly fast and robust implementations:

- Mallet by McCallum [**McCallum2002**], Java
- gensim by Řehůřek [**Rehurek2010**], Python

2.2 Word embeddings

In traditional language modeling, words are treated as atomic units by a discrete one-hot encoding. Given a vocabulary V, each word w is assigned a vector of length $|V|$ with all values set to zero except one to uniquely identify the word. A typical architecture involves n-grams, which try to predict the next word based on previous words. This approach has two disadvantages. First, it limits the context to an arbitrary range. When n is too small, only the immediate preceding words are taken into account, failing to capture long-range word dependencies. When n is too large, the *curse of dimensionality* becomes a problem: when representing a joint sequence of n words, there are $|V|^n$ potential combinations. As most n-grams are never observed, this leads to very sparse vectors. Second, this type of language modeling does not capture similarity between words and also does not capture information about the relationship between words. When using the one-hot encoding, both the words "house" and "houses" get a unique vector with no indication of similarity. If one would employ cosine distance or euclidean distance to assess similarity in the vector space, all words would be equally similar.

In contrast to this, the idea of *distributed representations* is to not represent objects by discrete counts but rather by continuous points in a vector space $\mathbb{R}^m$, where m is much smaller than $|V|$. The idea of distributed representations has a long history, dating back to the mid-eighties with work from Hinton and Rumelhart [**Hinton1986**, **Rumelhart1988**]. Intuitively, instead of using one dimension for each atomic unit, distributed representations assign meanings to the dimensions and then recombine these meanings for a specific object. Recently, the term *word embeddings* was also used to describe the approach of embedding words in a vector space. We will use the terms *distributed representation*, *word vectors* and *word embeddings* interchangeably in this document.

The use of distributed representations for words was proposed by Bengio et al. [**Bengio2003**]. In their work, they train a neural network to predict a word given the preceding context. This prediction model is again based on the distributional hypothesis: words are similar, if they occur in the same contexts. With this architecture, "the system naturally learns to assign similar vectors to similar words" [**Baroni2014**].

In a series of papers, Mikolov et al. [**Mikolov2013**, **Mikolov2013a**, **Mikolov2013b**] provided a better neural architecture to train the word vectors. Contrary to the trend of ever deeper models, a shallower model with no hidden layer was more successful in building better word vectors, because it can be trained on more data. As Mikolov et al. [**Mikolov2013a**] state, "it might not be able to represent data as precisely as neural networks, but can possibly be trained on much more data efficiently." Due to the state-of-the-art performance and the free open source implementation *word2vec*[1] this method gained popularity and will also be used in this thesis.

Mikolov et al. presented two models. The continuous bag-of-words model (CBOW) works by predicting the center word in a symmetric context window. The continuous skip-gram model works the other way round: given a word, it tries to predict the context. We focus on the skip-gram model in the following, as it had the better evaluation results in the original paper. In the skip-gram model, surrounding words have to be encoded in the word vector for each word or as Mikolov et al. [**Mikolov2013**] state "vectors can be seen as representing the distribution of the context in which a word appears".

A nice property of the trained word vectors is, that linear relationships between words are kept in the vector space. It was shown that

$$vector(''King'') - vector(''Man'') + vector(''Woman'')$$

is closest to the vector representation of the word "queen" [**Mikolov2013b**]. Interestingly, this relationship holds for both semantic and syntactic similarly [**Mikolov2013a**], such as

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

$$vector(\text{"}houses\text{"}) - vector(\text{"}house\text{"}) + vector(\text{"}car\text{"})$$

is closest to $vector(\text{"}cars\text{"})$.

Word embeddings have been used in many natural language and machine learning tasks. They are a natural input to many machine learning algorithms, as they provide a better encoding than the traditional one-hot encoding. Concrete use cases involve statistical language modeling, machine translation [**Zou2013**], sentiment analysis [**Maas2011**] and paraphrase detection.

A word vector can be seen as a summary of the word with all its meanings. Many applications require not only word vectors, but also summaries of longer chunks of text. Therefore, it is necessary to embed sentences or documents in the word embedding space. For this problem, Le and Mikolov [**Le2014**] proposed paragraph vectors. In their work, paragraph can mean an arbitrary long text, ranging from sentences to entire documents. Paragraph vectors are trained by adding an artificial word to each context, which represents the current paragraph. These paragraph vectors can then be viewed as a summary of the meaning of the entire paragraph, which is useful for further downstream applications.

Implementations The original implementation of the approach by Mikolov et al. was open sourced under the name *word2vec*. It is implemented in C and can be used as a command line tool. There also exists another implementation in the gensim package by Řehůřek [**Rehurek2010**]. These two implementations can run multi-threaded in the CPU. There also exist some GPU implementations, however, these are not as stable. Therefore, we will concentrate on the CPU implementations first.

2.3 Comparison between LDA and word embeddings

Both LDA and word embeddings can provide distributed representations for words. In LDA, this works 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 [**Mikolov2013b**, **Mikolov2013a**]. Also, it tends to yield sparse vectors as LDA tries to keep the number of topics per word small.

Baroni et al. [**Baroni2014**] also introduce the 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 1 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 as being used 20 % in *sports* and 80 % in *politics*. When working with word embeddings, a vector like $[-2.4\ 0.3\ 1.3\ -0.1]$ is not interpretable. The arbitrary dimensions and values of a word embedding vector cannot be understood by humans.

Regarding the performance, LDA operates much slower than word embeddings. LDA becomes very expensive on large data sets, because it needs to repeatedly iterate over the entire corpus. The method by Mikolov has been successfully trained on a corpus with about 100 billion words.

2.4 Existing combinations

The existing related work, which combines topic modeling approaches with word embeddings, can be roughly divided in two directions. One type of model tries to improve topic models by incorporating word vectors, while the other aims to improve word embeddings with topic models. We summarize the prior work in this field in Table 2.1.[2]

The work by Das et al. [**Das2015**] belongs to the first group. They propose a method named 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. Also, words are no longer atomic units, but are represented by their corresponding pre-trained word embeddings. Each topic is associated with a mean μ_k and a covariance matrix Σ_k. The prior distribution over the means is a normal distribution, and the prior distribution over the covariances is an inverse Wishart distribution. These parameters are inferred using Gibbs sampling as in standard LDA. Another idea is proposed by Sridhar [**Sridhar2015**]. They start by training word embeddings. Afterwards they fit a Gaussian mixture model directly on the word embedding space. A topic is represented by mean and covariance matrix. In contrast to Das et al. [**Das2015**], Sridhar [**Sridhar2015**] assumes the topics to be fixed given the word embeddings, while Das et al. [**Das2015**] only take the embeddings as a basis and still learn the topics based on the word co-occurrences in the corpus. Another approach is taken by Nguyen et al. [**Nguyen2015**]. They 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 $s \sim Ber(\lambda)$. This parameter decides, whether the word is sampled

[2]ToCoh: Topic Coherence; WordSim: Word Similarity; DocClass: Document Classification; DocClust: Document Clustering; Analogy: Word Analogy Reasoning; SentComp: Sentence Completion

[3]The online evaluation tool can be accessed at http://palmetto.aksw.org/palmetto-webapp

Table 2.1: Overview over related work of combinations between topic models (TM) and word embeddings (WE) and how they have been evaluated

Paper	**TM**	**WE**	**ToCoh**	**WordSim**	**DocClass**	**DocClust**	**A**
Blei et al. [**Blei2003**]	✓				✓ REUTERS		
Mikolov et al. [**Mikolov2013a**]		✓					
Das et al. [**Das2015**]	✓		✓ Wikipedia co-occurrence				
Nguyen et al. [**Nguyen2015**]	✓		✓ Wikipedia co-occurrence		✓ 20-NEWS-GROUP, TAGMYNEWS	✓ Same as left	
Sridhar [**Sridhar2015**]	✓		✓ Manual annotation				
Moody [**Moody2016**]	✓	✓	✓ Palmetto[3] evaluation tool	✓ Only qualitative			
Liu et al. [**Liu2015**]		✓		✓ SCWS	✓ 20-NEWS-GROUP		
Cheng et al. [**Cheng2015**]		✓		✓ SCWS + paraphrase detection MSRPC			

as usual from the topic-word distribution or from the latent feature vectors. To sample in the embedding space, they sample from a softmax-normalized categorical distribution defined by:

$$Cat(w|\tau_t, \omega) = \frac{exp(\tau_t * \omega_w)}{\sum_{w' \in W} exp(\tau_t * \omega_{w'})},$$

where τ_t is the representation of topic t in the word embedding space and ω are the pretrained and fixed word embedding vectors. The authors use pretrained vectors from *word2vec* [**Mikolov2013a**] and *GloVe* [**Pennington2014**]. The topic embeddings τ are updated in each iteration of Gibbs sampling via a MAP estimate based on all other parameters given. A similar idea is used by Cheng et al. [**Cheng2015**]. In fact, the first model they propose is the same as the first model from Nguyen et al. [**Nguyen2015**]. In addition to some other new models, which are similar to Cheng et al.'s work, they also propose new models, which do not assume independence between word and concept embeddings. These *generative word-concept skip-gram* models perform best in their word-similarity and paraphrase detection benchmarks.

We now focus on methods for better word embeddings. Liu et al. [**Liu2015**] propose topical word embeddings. These embeddings make use of word topics inferred via LDA to generate better word vectors. They propose three different architectures. The first method regards topic as pseudo words, which are added to the context of each word. Thus, an embedding for a word and its topic is learned independently. The second method considers each pair of word and topic as a pseudo word. The third method works the same way, except that it learns only one word embedding vector per word and one topic embedding vector per topic. These two vectors are then combined for the vector of the word-topic pair. To the authors' surprise, the simplest method one performs best in their word similarity and text classification benchmarks. Using the additional topic information when training word embeddings has practical benefits for similarity evaluation. When the model is asked for similar words to the word "apple", it returns "peach", "juice", "strawberry" or it returns "mac", "ipod", "android" depending on the topic of the word.

Moody [**Moody2016**] proposes lda2vec. His method learns word vectors together with learning topic vectors and an intuitive topic-based document interpretation. Even though trained in the word embedding space, the topic-document vectors are still kept parse to be interpretable for humans. However, the method is not evaluated extensively and it is unclear, whether this method yields improved word vectors.

Appendices