UNIVERSITY OF CALIFORNIA, SAN DIEGO

Interactive Topic Modeling

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Quentin Pleple

Committee in charge:

Professor Charles Elkan, Chair
Professor Sanjoy Dasgupta
Professor Mohan Paturi

2013

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Charles Elkan for the always relevant feedback and useful discussions. Furthermore, I would like to thank Rakesh Varna for his unflagging and continuous support, and Russell Reas for being a solid partner. I am finally grateful to Christopher Tosh for his encouragement, Matus Telgarsky for his insightful comments and support.

ABSTRACT OF THE THESIS

Interactive Topic Modeling

by

Quentin Pleple

Master of Science in Computer Science

University of California, San Diego, 2013

Professor Charles Elkan, Chair

Topics discovered by the latent Dirichlet allocation (LDA) method are sometimes not meaningful for humans. The goal of our work is to improve the quality of topics presented to end-users. Our contributions are two-fold. First, we present a new way of picking words to represent a topic. Instead of simply selecting the top words by frequency, by penalizing words that are shared across multiple topics, we down-weight background words and reveal what is specific about each topic. Second, we present a novel method for interactive topic modeling. The method allows the user to give live feedback on the topics, and allows the inference algorithm to use that feedback to guide the LDA parameter search. The user can indicate that words should be removed from a topic, that topics should be merged, and/or that a topic should be split, or deleted. After each item of user feedback, we change the internal state of the variational EM algorithm in a way that preserves correctness, then re-run the algorithm until convergence. Experiments show that both contributions are successful in practice.

Chapter 1

Background

1.1 Bayesian inference

Suppose we have a probabilistic model with observations $\boldsymbol{x}$ and hidden variables $\boldsymbol{z}$. We want to infer the hidden variables $\boldsymbol{z}$ that can best explain the observations $\boldsymbol{x}$. In the Bayesian framework, we treat parameters as hidden random variables as well.

Maximum Likelihood. The common way to perform inference is to compute the Maximum Likelihood estimate

$$\hat{\boldsymbol{z}}_{\text{ML}} = \underset{\boldsymbol{z}}{\operatorname{argmax}}\, p(\boldsymbol{x}|\boldsymbol{z})$$

which is the set of hidden variables that maximizes the likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ of the observed data.

Maximum a posteriori. A Bayesian approach will assume some prior knowledge about the hidden variables: a distribution $p(\boldsymbol{z})$ over the space of hidden variables $\boldsymbol{z}$. The likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ gets weighted by the prior knowledge $p(\boldsymbol{z})$ to give the posterior $p(\boldsymbol{z}|\boldsymbol{x})$:

$$\text{posterior} \ \propto \ \text{likelihood} \ \times \ \text{prior} \qquad \text{or here} \qquad p(\boldsymbol{z}|\boldsymbol{x}) \ \propto \ p(\boldsymbol{x}|\boldsymbol{z}) \ \times \ p(\boldsymbol{z}).$$

The maximum a posteriori (MAP) estimate

$$\hat{\boldsymbol{z}}_{\text{MAP}} = \underset{z}{\operatorname{argmax}}\, p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}) = \underset{z}{\operatorname{argmax}}\, p(\boldsymbol{z}|\boldsymbol{x})$$

is the set of hidden variables that maximizes the posterior. The main advantage of this approach is that in order to maximize $p(\boldsymbol{z}|\boldsymbol{x})$, we don't need to compute the normalizing constant

$$p(\boldsymbol{x}) = \int_{\boldsymbol{z}} p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})\mathrm{d}\boldsymbol{z}$$

whereas it is needed for the full Bayesian approach, as we will see in the next paragraph. This normalizer can become intractable when models get complicated. The MAP method is therefore sometimes described as poor man's Bayesian inference (Tzikas et al., 2008) as this is a way of including prior knowledge without having to pay the expensive price of computing $p(\boldsymbol{x})$.

Full Bayesian approach. In the full Bayesian approach, we don't only want one estimate $\hat{\boldsymbol{z}}$ of the hidden variables, but the entire distribution over them, the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$.

Every time we pick one specific value for a parameter in a model, we are making an approximation. As we compose models, approximations get amplified at every layer of the model. Feeding the entire parameters distribution in the next layer, instead of a point estimate, will increase the added value of this layer to the model.

In this approach, the normalizer $p(\boldsymbol{x})$ becomes intractable as the model gets more complex, making it impossible to compute the posterior distribution. Often, the best we can do is approximate the posterior.

1.2 Approximation methods for Bayesian inference

There are two method families to approximate intractable posterior distributions: deterministic methods such as variational inference, and stochastic methods such as sampling methods.

Deterministic methods. Variational methods come from the mathematical field calculus of variations where the goal is to find the function that optimizes a given numerical quantity.

One common variational method is Variational Expectation-Maximization (EM). Standard EM is used to get Maximum Likelihood estimates

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z}}{\operatorname{argmax}}\, p(\boldsymbol{x}|\boldsymbol{z}).$$

But when the model gets complicated, computing the posterior $p(\boldsymbol{z}|\boldsymbol{x})$ in the E step becomes computationally intractable. Variational Bayes approximates the posterior by removing some dependencies between some variables of the model to get a tractable distribution $q(\boldsymbol{z})$ over the space of hidden variables $\boldsymbol{z}$. Variational EM approach approximates the posterior by considering a parameterized family of tractable distributions (Bishop, 2006).

Stochastic methods. Gibbs sampling is a Markov Chain Monte Carlo algorithm (Geman and Geman, 1984) that repeatedly picks one hidden variable z_i at random and samples it from the distribution of that variable conditioned on all other hidden variables $\boldsymbol{z}_{-i}$ and the observation $\boldsymbol{x}$.

Algorithm 1 Gibbs sampling

```
Initialize $\boldsymbol{z}$ randomly
repeat
    Pick $i$ randomly
    Draw $z_i \sim p(z_i|\boldsymbol{z}_{-i}, \boldsymbol{x})$
until convergence
```

We refer to convergence loosely here, as Gibbs sampling doesn't converge and will eventually visit all possible state, maybe in an exponential number of iterations. We'll usually assume convergence when the state is not changing too much over a reasonable number of iterations.

1.3 Linear topic modeling

The interest in learning some kind of topics from a corpus of documents started from the publication of Latent Semantic Analysis (LSA) (Deerwester et al., 1990), also called Latent Semantic Indexing (LSI) in the context of information retrieval. LSA is a linear method based on the factorization of the document-word matrix $\boldsymbol{X}$, where x_{dw} is the count of occurrences of word w in document d. The goal is to find a low-rank approximation $\tilde{\boldsymbol{X}}$ of $\boldsymbol{X}$ factorizing it into two matrices, one representing the documents, and the other the topics. In this model, documents are converted to the bag-of-words format which ignores the ordering of words and only keeps the word counts.

Singular Value Decomposition. The most common way is to use the Singular Value Decomposition (SVD) of

$$\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal matrices and Σ is diagonal. By selecting only the K largest singular values from Σ and the corresponding vectors in $\boldsymbol{U}$ and $\boldsymbol{V}^T$, we get the best rank K approximation of matrix $\boldsymbol{X}$ according to the loss $\|\boldsymbol{X} - \tilde{\boldsymbol{X}}\|_2^2$. Rows of $\boldsymbol{U}$ represent documents in a K-dimensional space, and columns of $\boldsymbol{V}^T$ represents the topics in the same space. Each document can be expressed as a linear combination of topics.

Non-negative Matrix Factorization. Another common approach is to use use Non-negative Matrix Factorization (NMF) on the document-word matrix $\boldsymbol{X}$ (Lee and Seung, 1999; Pauca et al., 2004). Here, $\tilde{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{V}$ where $\boldsymbol{U}$ represents the topics and $\boldsymbol{V}$ the documents, both being non-negative matrices.

1.4 Probabilistic topic modeling

LSA represents topics as points in Euclidean space and documents as linear combination of topics. Probabilistic topic models differ from LSA by representing topics as distributions over words, and documents as probabilistic mixtures of topics. Formally, given a vocabulary of W words, each topic $k = 1, ..., K$ has a word-distribution $\boldsymbol{\varphi}_k \in \Delta_W$, with Δ_n the simplex of dimension n, and each document $d = 1, ..., D$ has a topic-distribution $\boldsymbol{\theta}_d \in \Delta_K$. Call $\boldsymbol{\Phi}$ the matrix with rows $\boldsymbol{\varphi}_k$ and $\boldsymbol{\Theta}$ the matrix with rows $\boldsymbol{\theta}_d$. Table 1.1 gives the list of symbols we will be using.

Table 1.1: Definition of symbols

Symbol	Definition
$\Delta_n \subset \mathbb{R}^n$	Simplex of dimension n
$W \in \mathbb{N}$	Number of words in the vocabulary
$D \in \mathbb{N}$	Number of documents in the corpus
$K \in \mathbb{N}$	Number of topics
$N_d \in \mathbb{N}$	Number of words in document d
$n_{dw} \in \mathbb{N}$	Count of word w in document d
$w_{di} \in [\![1, W]\!]$	Word at position i in document d
$z_{di} \in [\![1, K]\!]$	Topic assignment of the word w_{di}
$\alpha \in \mathbb{R}^+$	Parameter of a symmetric Dirichlet prior over $\boldsymbol{\theta}_d$
$\beta \in \mathbb{R}^+$	Parameter of a symmetric Dirichlet prior over $\boldsymbol{\varphi}_k$
$\boldsymbol{\psi}_{dw} \in \Delta_K$	Topic-distribution for word w in document d
$\boldsymbol{\varphi}_k \in \Delta_W$	Word-distribution for topic k
$\boldsymbol{\theta}_d \in \Delta_K$	Topic-distribution for document d
$\boldsymbol{\lambda}_k \in \mathbb{R}^W$	Parameter of a Dirichlet prior over $\boldsymbol{\varphi}_k$ in the variational setup
$\boldsymbol{\gamma}_d \in \mathbb{R}^K$	Parameter of a Dirichlet prior over $\boldsymbol{\theta}_d$ in the variational setup
$\boldsymbol{\Phi} \in \mathbb{R}^{W \times K}$	Matrix of rows $\boldsymbol{\varphi}_k$
$\boldsymbol{\Theta} \in \mathbb{R}^{K \times D}$	Matrix of rows $\boldsymbol{\theta}_d$
$\boldsymbol{\lambda} \in \mathbb{R}^{W \times K}$	Matrix of rows $\boldsymbol{\lambda}_k$
Ψ	Digamma function

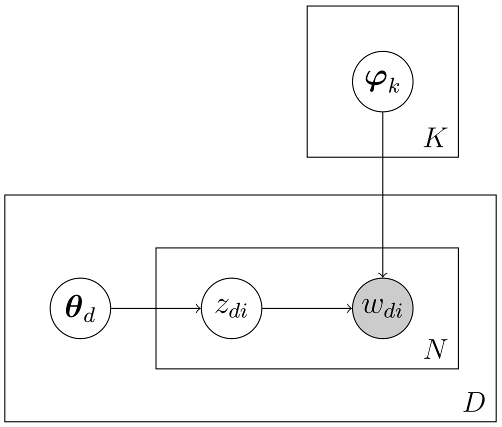


Figure 1.1: pLSI model

By estimating the model parameters $(\boldsymbol{\Phi}, \boldsymbol{\Theta})$ of the model, we discover new knowledge about the corpus; topics are represented by the word-distributions $\boldsymbol{\varphi}_k$, and each document is tagged with discovered topics, which is represented by $\boldsymbol{\theta}_d$.

Probabilistic Latent Semantic Indexing (pLSI) is described as a generative process (Hofmann, 1999), a procedure that probabilistically generates documents given the parameters of the model (Algorithm 2). Parameters of the model are then learned by Bayesian inference.

Algorithm 2 Generative process for pLSI

```
procedure GENERATIVEPLSI(document-tagging $\boldsymbol{\theta}_d$, topics $\boldsymbol{\varphi}_k$)
    for document $d = 1, ..., D$ do
        for position $i = 1, ..., N_d$ in document $d$ do
            Draw a topic $z_{di} \sim$ Discrete($\boldsymbol{\theta}_d$)
            Draw a word $w_{di} \sim$ Discrete($\boldsymbol{\varphi}_{z_{di}}$)
    return counts $n_{dw} = \sum_i I(w_{di} = w)$
```

Although the formulation of the model is probabilistic, Ding et al. (2008) proved the equivalence between pLSI and NMF, by showing that they both optimize the same objective function. As they are different algorithms, they will navigate in the parameters space differently. It is possible to design an hybrid algorithm alternating between NMF and pLSI, every time jumping out of the local optimum of the other method.

1.5 Latent Dirichlet Allocation

Blei et al. (2003) extended pLSI by adding a symmetric Dirichlet prior Dir(α) on topic-distributions $\boldsymbol{\theta}_d$ of documents and derived a variational EM algorithm for the Bayesian inference. Griffiths and Steyvers (2002a,b) went further by adding[1] a symmetric Dirichlet prior Dir(β) on topics $\boldsymbol{\varphi}_k$, and derived a Gibbs sampler for the Bayesian inference. The generative process of LDA is described in Algorithm 3.

Algorithm 3 Generative process for LDA

```
procedure GENERATIVELDA(document-tagging smoothing $\alpha$, topic smoothing $\beta$)
    for topic $k = 1, ..., K$ do
        Draw a word-distribution $\boldsymbol{\varphi}_k \sim$ Dir$(\beta)$
    for document $d = 1, ..., D$ do
        Draw a topic-distribution $\boldsymbol{\theta}_d \sim$ Dir$(\alpha)$
        for position $i = 1, ..., N_d$ in document $d$ do
            Draw a topic $z_{di} \sim$ Discrete$(\boldsymbol{\theta}_d)$
            Draw a word $w_{di} \sim$ Discrete$(\boldsymbol{\varphi}_{z_{di}})$
    return counts $n_{dw} = \sum_i I(w_{di} = w)$
```

Bayesian model can also be described graphically in plate notation which helps to understand the dependencies between random variables. The graphical representation of LDA is presented in Figure 1.2.

We suppose documents are generated according to this generative model and we want to estimate values for a set a parameters $(\boldsymbol{\Phi}, \boldsymbol{\Theta})$ that can best explain the set of observations: the word counts n_{di}.

Theoretically, we could learn hyperparameters α and β using Newton-Raphson method (Blei et al., 2003). But usually, hyperparameters are fixed heuristically to simplify the algorithm and make it converge faster. Common values are $\alpha = \frac{1}{K}$ and $\beta = 0.1$ (Steyvers

[1]Chronologically, Blei et al. (2002) first published a paper presenting LDA in *NIPS* treating topics $\boldsymbol{\varphi}_k$ as free parameters. Shortly after, Griffiths and Steyvers (2002a,b) extended this model by adding a symmetric Dirichlet prior on $\boldsymbol{\varphi}_k$. Finally, Blei et al. (2003) published an extended version their first paper in *Journal of Machine Learning Research* (by far the most cited LDA paper) with a section on having this Dirichlet smoothing on multinomial parameters $\boldsymbol{\varphi}_k$.

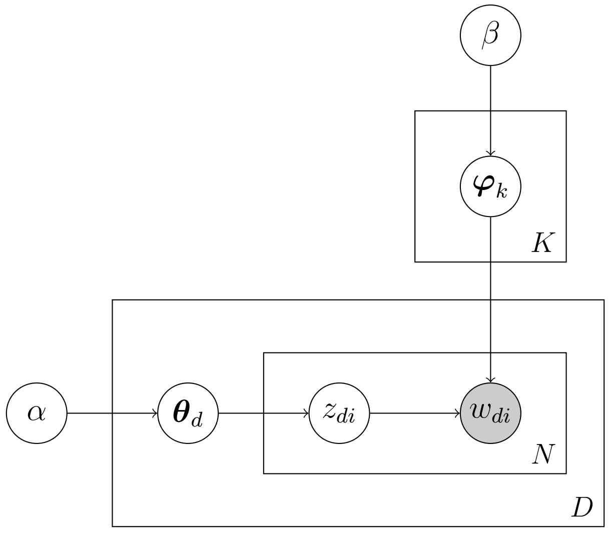


Figure 1.2: LDA model

and Griffiths, 2006).

Chapter 2

Evaluating topic models

Despite the intensive work on topic models during the last decade, we still don't have any convincing way of evaluating their goodness of fit and this is still an open research question (Blei, 2012).

Our work deals with improving the topics found by LDA, either by changing their representation (Chapter 3), or by allowing the user to give live guidance to the inference algorithm (Chapter 4). In order to measure these improvements, we need to be able to evaluate a given output of LDA. This chapter presents the state-of-the-art in terms of automatic ways of measuring topic usefulness.

2.1 Perplexity

The most common way to evaluate a probabilistic model is to measure the log-likelihood of a held-out test set. This is usually done by splitting the dataset into two parts: one for training, the other for testing. For LDA, a test set is a collection of unseen documents $\boldsymbol{w}_d$, and the model is described by the topic matrix $\boldsymbol{\Phi}$ and the hyperparameter α for topic-distribution of documents. The LDA parameters $\boldsymbol{\Theta}$ is not taken into consideration as it represents the topic-distributions for the documents of the training set, and can therefore be ignored to compute the likelihood of unseen documents. Therefore, we need to evaluate the log-likelihood

$$\mathcal{L}(\boldsymbol{w}) = \log p(\boldsymbol{w}|\boldsymbol{\Phi}, \alpha) = \sum_d \log p(\boldsymbol{w}_d|\boldsymbol{\Phi}, \alpha).$$

of a set of unseen documents $\boldsymbol{w}_d$ given the topics $\boldsymbol{\Phi}$ and the hyperparameter α for topic-distribution $\boldsymbol{\theta}_d$ of documents. Likelihood of unseen documents can be used to compare models; higher likelihood implying a better model.

The measure traditionally used for topic models is the *perplexity* of held-out documents $\boldsymbol{w}_d$ defined as

$$\text{perplexity}(\text{test set } \boldsymbol{w}) = \exp\left\{-\frac{\mathcal{L}(\boldsymbol{w})}{\#\text{ tokens}}\right\}$$

which is a decreasing function of the log-likelihood $\mathcal{L}(\boldsymbol{w})$ of the unseen documents $\boldsymbol{w}_d$; the lower the perplexity, the better the model.

However, the likelihood $p(\boldsymbol{w}_d|\boldsymbol{\Phi}, \alpha)$ of one document is intractable, which makes the evaluation of $\mathcal{L}(\boldsymbol{w})$, and therefore the perplexity, intractable as well. Wallach et al. (2009) derive various sampling methods to approximate this probability.

2.2 Perplexity is not strongly correlated to human judgment

Chang et al. (2009) have shown that, surprisingly, predictive likelihood (or equivalently, perplexity) and human judgment are often not correlated, and even sometimes slightly anti-correlated.

They ran a large scale experiment on the Amazon Mechanical Turk platform. For each topic, they took the top five words (ordered by frequency $p(w|k) = \varphi_{kw}$) of that topic and added a random sixth word. Then, they presented these lists of six words to participants asking them to identify the intruder word.

If every participant could identify the intruder, then we could conclude that the topic is good at describing an idea. If on the other hand, many people identified one of the topic top five word as the intruder, it means that they could not see the logic in the association of words, and we can conclude the topic was not good enough.

It's important to understand what this experiment is proving. The result proves that, given a topic, the five words that have the largest frequency $p(w|k) = \varphi_{kw}$ withing their topic are usually not good at describing one coherent idea; at least not good enough to be able to recognize an intruder.

2.3 Modeling human judgment

Human judgment not being correlated to perplexity (or likelihood of unseen documents) is the motivation for more work trying to model the human judgment. This is by itself a hard task as human judgment is not clearly defined; for example, two experts can disagree on the usefulness of a topic.

One can classify the methods addressing this problem into two categories. *Intrinsic* methods that do not use any external source or task from the dataset, whereas *extrinsic* methods use the discovered topics for external tasks, such as information retrieval (Wei and Croft, 2006), or use external statistics to evaluate topics.

As an early intrinsic method, AlSumait et al. (2009) define measures based on three prototypes of junk and insignificant topics. The three prototypes for junk topics are the uniform word-distribution, the empirical corpus word-distribution, and the uniform document-distribution:

$$p(w|\text{topic}) \propto 1 \qquad p(w|\text{topic}) \propto \text{count}(w \text{ in corpus}) \qquad p(d|\text{topic}) \propto 1$$

Then a topic significance score is computed from various dissimilarities and similarities (KL divergence, cosine, and correlation) to these three prototypes. However, the significance score is a complicated function with free parameters, that seem to be arbitrarily chosen, so the risk of overfitting the two datasets used for experiments is high.

2.3.1 Topic coherence

The state-of-the-art in terms of topic coherence are the intrinsic measure UMass and the extrinsic measure UCI, both based on the same high level idea. Both measure compute the sum

$$\text{Coherence} = \sum_{i<j} \text{score}(w_i, w_j)$$

of pairwise scores on the words w_1, ..., w_n used to describe the topic, usually the top n words by frequency $p(w|k)$. This measure can be seen as the sum of all edges on the graph shown in Figure 2.1.

Notation. Let's define $D(w_i)$ as the count of documents containing the word w_i, $D(w_i, w_j)$ the count of documents containing both words w_i and w_j, and D the total number or documents in the corpus. The corpus used to compute the counts is specified in a subscript of symbol D. For example, $D_{\text{Wikipedia}}(w_i)$ it the count of documents of the Wikipedia corpus containing the word w_i. When no subscript is specified, the corpus used is the corpus on which the model have been trained.

2.3.2 Extrinsic UCI measure

The UCI measure introduced by Newman et al. (2010) uses as pairwise score function the Pointwise Mutual Information (PMI)

$$\text{score}_{\text{UCI}}(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}$$

where $p(w)$ represents the probability of seeing w_i in a random document, and $p(w_i, w_j)$ the probability of seeing both w_i and w_j co-occurring in a random document. Those probabilities are empirically estimated from an external dataset such as Wikipedia:

$$p(w_i) = \frac{D_{\text{Wikipedia}}(w_i)}{D_{\text{Wikipedia}}} \qquad \text{and} \qquad p(w_i, w_j) = \frac{D_{\text{Wikipedia}}(w_i, w_j)}{D_{\text{Wikipedia}}}.$$

Given the score function, we are free to choose the corpus to compute the empirical probabilities. Newman et al. (2010) chose three external corpus to evaluate them (Wikipedia, Google 2-grams, and Medline) but not the corpus that generated the topics. The argument given is that using the same dataset would reinforce noise or unusual word statistics. However, some intrinsic topic coherence measures have been developed since, that are also better correlated to human judgment than perplexity (Mimno et al., 2011) (see next section). It may be worth comparing intrinsic and extrinsic PMI-based measures.

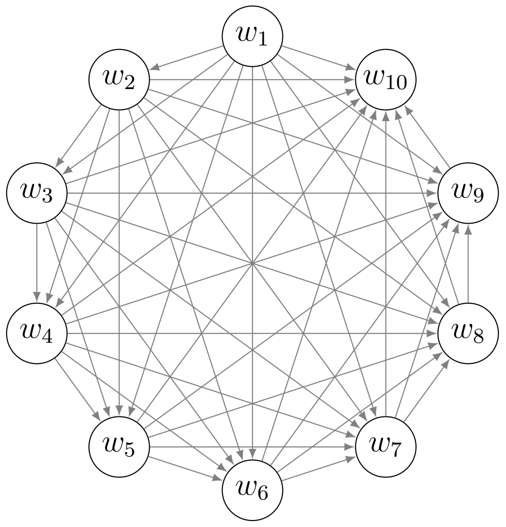


Figure 2.1: Both topic coherence measures UCI and UMass are based on the sum $\sum_{i<j}$ score(w_i, w_j) of the pairwise scores of the n top words w_1, ..., w_n of the topic.

2.3.3 Intrinsic UMass measure

The UMass measure introduced by Mimno et al. (2011) uses as pairwise score function

$$\text{score}_{\text{UMass}}(w_i, w_j) = \log \frac{D(w_i, w_j) + 1}{D(w_i)}$$

which is the empirical conditional log-probability $\log p(w_j|w_i) = \log \frac{p(w_i,w_j)}{p(w_j)}$ smoothed by adding one to $D(w_i, w_j)$.

The score function is not symmetric as it is an increasing function of the empirical probability $p(w_j|w_i)$, where w_i is more common than w_j, words being ordered by decreasing frequency $p(w|k)$. So this score measures how much, within the words used to describe a topic, a common word is in average a good predictor for a less common word.

As the pairwise score used by the UMass measure is not symmetric, the order of the arguments matters. UMass measure is computing p(rare word | common word), how much a common word triggers a rarer word. However, in human word association, high frequency words are more likely to be used as response words than low frequency words (Steyvers and Griffiths, 2006). It would be interesting to understand the effect of this choice by doing more experiments and comparing the two options.

2.4 On smoothing in topic coherence measures

Stevens et al. (2012) performed an extensive study of the these two measures on one dataset (New York Times articles[1]) in order to compare the three models: LDA, LSA with SVD, and LSA with NMF. However, they did not use the formulations of the measures used by their original authors.

For the UMass measure, they introduced a free parameter ε, instead of just one, for smoothing in the pairwise scoring function

$$\text{score}_{\text{UMass}}(w_i, w_j) = \log \frac{D(w_i, w_j) + \varepsilon}{D(w_i)}$$

and tried both $\varepsilon = 1$ and $\varepsilon = 10^{-12}$. Setting $\varepsilon = 10^{-12}$ seems to over-penalize pairs that never occur together, i.e. when $D(w_i, w_j) = 0$, as it will decrease the score of that pair by 12

$$\text{score}_{\text{UMass}}(w_i, w_j) = \log \frac{\varepsilon}{D(w_i)} = -12 - \log D(w_i)$$

which is very large in the log space of document counts. It is also equivalent to say that we would have to see 10^{12} more documents to see the two words appearing only once together. Having $\varepsilon = 1$ looks more reasonable as it is treating pairs that appear once throughout the corpus, i.e. when $D(w_i, w_j) = 1$, and pairs that never appear, i.e. when $D(w_i, w_j) = 0$, roughly in the same order of magnitude.

For the UCI measure, they introduced a smoothing ε in the pairwise score that the original authors did not

$$\text{score}_{\text{UCI}}(w_i, w_j) = \log \frac{p(w_i, w_j) + \varepsilon}{p(w_i)p(w_j)}$$

with ε initially set to one. Here, we are at a different scale as we are dealing with probabilities and not counts. As $\varepsilon = 1$ is likely to be huge compared to $p(w_i, w_j)$, the smoothing parameter artificially increases the topic coherence, and not even by the same amount for all pairs of words. Then, using $\varepsilon = 10^{-12}$, which is likely to be smaller than $p(w_i, w_j)$, caused big changes, and therefore the authors concluded that coherence measures depend heavily on smoothing ε.

[1]92,600 New York Times articles from 2003 and a vocabulary size of 35,836 tokens after removing the ones occurring less than 200 times throughout the corpus.

Nonetheless a smoothing parameter is required to avoid taking the logarithm of zero. A reasonable choice for smoothing is to assume that every pair of words is present at least once in the corpus, and compute the empirical probability

$$p(w_i, w_j) = \frac{D_{\text{Wikipedia}}(w_i, w_j) + 1}{D_{\text{Wikipedia}}}$$

which is the same smoothing as the UMass measure.

Chapter 3

Word relevance within a topic

The most common way to display a topic, a discrete distribution over words, is to print out the top ten words ordered by decreasing frequency within this topic. Given a single topic, there is nothing much more we can do. But knowing other topics that are describing the same corpus gives us more information. It seems we can use this information to pick better words to represent topics.

Chapter 2 presents the state-of-the-art in evaluation of topic coherence. Both presented metrics are the sum of pairwise score on the ten top words. In this chapter we present a novel way to pick words to represent topics. We illustrate it by examples of this new representation, and compare the change in topic coherence from the old to the new representation with the measures presented in the previous chapter.

3.1 Word distinctiveness and saliency

In order to find the best informative words of a corpus, Chuang et al. (2012) first define word *distinctiveness*

$$\mathcal{D}(w) = \sum_k p\,(k|w) \log \frac{p(k|w)}{p(k)} = \mathrm{KL}\big(p(k|w) \parallel p(k)\big)$$

of a word as the Kullback–Leibler (KL) divergence between, the topic distribution $p(k|w)$ given the word w, and the marginal topic distribution $p(k)$, the likelihood that any random word has been drawn from topic k. The word distinctiveness measures how much a word is shared across topics. The higher the distinctiveness, the less this word is shared across topics.

Then, they define the word *saliency*

$$\mathcal{S}(w) = p(w)\mathcal{D}(w)$$

of a word w by weighting its frequency by its distinctiveness. Compared to the ranking by frequency $p(w)$, the ranking by saliency $p(w)\mathcal{D}(w)$ will penalize the words shared across several topics, as they will have a low distinctiveness, and boost words that are good predictors of one topic, as they will have a high distinctiveness.

3.2 Word relevance for a topic

Word saliency and distinctiveness have been design to find relevant words corpus-wide, not for a specific topic. They are not good to find candidates for topic representatives. In this section, we present a word relevance score within a topic based on the same idea: penalize the word frequency by a factor that captures how much the word is shared across topics.

First, instead of the global word frequency $p(w)$, we consider the frequency $p(w|k)$ of the word within a topic k. Then as a sharing penalty, we divide by the exponential entropy e^{H_w}, where

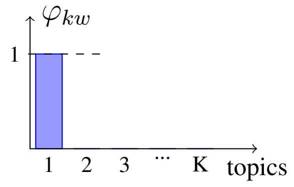
$$H_w \triangleq -\sum_k p(k|w) \log p(k|w)$$

is the entropy of the distribution of topics given a word w, capturing how much the word w is shared across several topics. We define the relevance measure

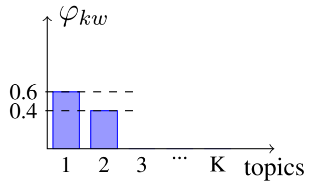
$$\mathcal{R}(w|k) \triangleq \frac{p(w|k)}{e^{H_w}}$$

for word w within topic k as being the frequency divided by the exponential entropy.

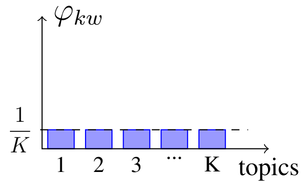
Interpretation of exponential entropy e^H. The exponential entropy can be seen as a measure of the extent, or the spread, of a distribution (Campbell, 1966). By extent, we mean the size of the support, or the number of elements with non-zero probability. Figure 3.1 illustrates it on three examples.



(a) Delta distribution
$H = 0$, and $e^H = 1$



(b) Example distribution
$H = 0.67$, and $e^H = 1.96$



(c) Uniform distribution
$H = \log K$, and $e^H = K$

Figure 3.1: These figures illustrate how the exponential entropy e^H measures the extent of a distribution. Distribution (a) has only one word as support, and $e^H = 1$. Distribution (b) is spanning only two words, but not uniformly so its exponential entropy is slightly below two. Distribution (c) is spanning uniformly K words, and $e^H = K$.

3.3 Computation of word relevance measure

Let's see how to express relevance $\mathcal{R}(w|k)$ in terms of LDA parameters. The numerator is straightforward

$$\mathcal{R}(w|k) \triangleq \frac{p(w|k)}{e^{H_w}} = \frac{\varphi_{kw}}{e^{H_w}}$$

Now, computing the entropy

$$E_w = \sum_k p(k|w) \log p(k|w)$$

requires applying Bayes rule on $p(k|w)$:

$$\begin{aligned} p(k|w) &\propto p(w|k)\; p(k) \\ &= \varphi_{kw} \sum_d p(k|d)p(d) \\ &\propto \varphi_{kw} \sum_d \theta_{dk} N_d \end{aligned}$$

where N_d is the length of document. Recombining the results, we get a procedure to compute the relevance score:

(a) Compute topic-distribution given word w:

$$p(k|w) \propto \varphi_{kw} \sum_d \theta_{dk} N_d$$

(b) Compute its entropy:

$$H_w \triangleq \sum_k p(k|w) \log p(k|w)$$

(c) Divide the frequency of word w within topic k by the exponential entropy:

$$\mathcal{R}(w|k) \triangleq \frac{p(w|k)}{e^{H_w}}$$

3.4 Top relevant word to describe topics

When running LDA on a corpus, some *background* words are going to be frequent throughout the corpus, and therefore be found as top words by frequency $p(w|k)$ of several topics. Table 3.1 shows this situation on the [20news] corpus where most of the top words are background words and don't convey any meaning: *people, writes, article, good*, etc. Those background words are shared across numerous topics, and their e^{H_w} will be high. They will be penalized by e^{-H_w} and score low on word relevance measure. On the other hand, words scoring high on word relevance measure $p(w|k)e^{-H_w}$ are more descriptive, and topics that did not make sense when described by top words by frequency $p(w|k)$, become intelligible when described by top words by relevance $p(w|k)e^{-H_w}$.

Table 3.1: Three topics extracted from the output of LDA run on the corpus [20news] with 30 topics. For each topic, we give the top words according to frequency $p(w|k)$ and relevance $p(w|k)e^{-H_w}$.

	Topic 1	Topic 2	Topic 3
Top words by frequency $p(w\|k)$	*people* *writes* *article* *guns* *police* *government* *state*	*writes* *space* *article* *power* *radio* *ground* *problem*	*writes* *article* *good* *cars* *engine* *bike* *time*
Top words by relevance $p(w\|k)e^{-H_w}$	*guns* *firearms* *weapons* *firearm* *handgun* *crime* *police*	*voltage* *circuit* *space* *larson* *wiring* *circuits* *wire*	*engine* *cars* *bike* *tires* *drive* *miles* *ford*

This effect is even stronger for specialized corpora, such as [nips] or [nsf] that contain research papers of one specific area. Table 3.2 shows six topics extracted from a run of LDA on the [nips] corpus. Background words here are not general English words, like for [20news], but words describing the field of the research papers: *network, networks, neural*, and *input*. Each of these topics, taken alone by itself, will look intelligible by humans. However, all presented next to each other, it becomes unclear what is the difference between them. Being able to understand interactions between topics is important when topics are used for an external task, such as browsing the corpus or information retrieval.

Showing the top words by relevance $p(w|k)e^{-H_w}$ will show what is specific about each topic. Topic 1 is more about chips and circuits where topic 3 is specifically about head direction cells, a special type of neurons involved in self-motion, and topic 4 about controlled substances (drugs, precursors and plants).

Table 3.2: Six topics extracted from the output of LDA run on the highly specialized corpus [nips] with 30 topics. For each topic, we give the top words by frequency $p(w|k)$ and relevance $p(w|k)e^{-H_w}$.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
Top words	*network*	*network*	*network*	*network*	*network*	*network*
	input	*neural*	*input*	*neural*	*networks*	*neural*
	output	*training*	*model*	*control*	*neural*	*networks*
sorted by	*neural*	*networks*	*system*	*model*	*neurons*	*algorithm*
$p(w\|k)$	*time*	*input*	*neural*	*system*	*dynamics*	*time*
Relevant	*chip*	*characters*	*head*	*controller*	*dynamics*	*dataflow*
words	*analog*	*character*	*cells*	*precursor*	*neurons*	*network*
	network	*network*	*motor*	*plant*	*bifurcation*	*neural*
sorted by	*circuit*	*printed*	*network*	*parse*	*neuron*	*networks*
$\mathcal{R}(w\|k)$	*voltage*	*classifier*	*vestibular*	*control*	*network*	*boolean*

3.5 Coherence of topics described by top words by relevance

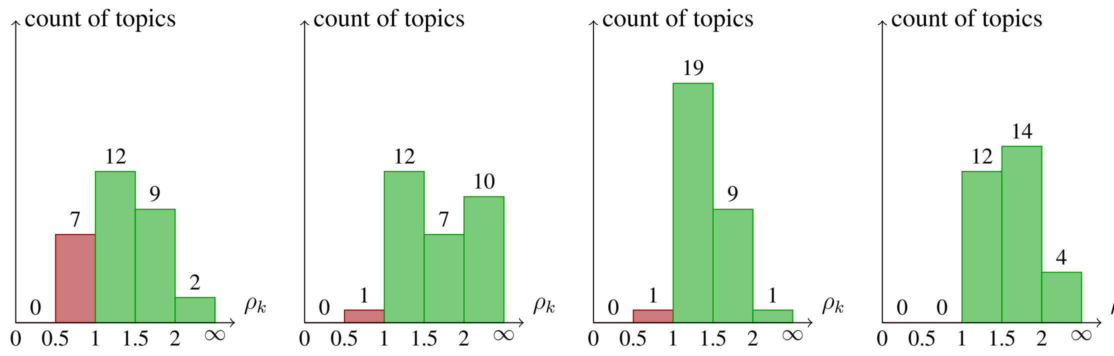
We have seen in the previous section a couple of examples where indeed showing words scoring higher on relevance are better candidates to describe topics than top words by frequency. Now, there are some automated measures of topic coherence, and we have presented the state-of-the-art in Chapter 2; it consists of the intrinsic measure UMass based on the conditional probability of one word given another, and the extrinsic measure UCI based on the the Pointwise Mutual Information (PMI) of words, computed from an external corpus such as Wikipedia. In this section, we compare the change in topic coherence according to these two metrics when changing the topic representation from top words by frequency to top words by relevance.

Wikipedia as the external corpus. To compute the UCI measure topic coherence score, we chose Wikipedia as external corpus as it is large enough so that every pairs of common enough words will appear in at least one document. Wikipedia was also the external corpus showing best correlation between UCI coherence score and human judgment (Newman et al., 2010). To compute the document counts $D_{\text{Wikipedia}}(w_i, w_j)$ and $D_{\text{Wikipedia}}(w_i)$ needed for the topic coherence, we indexed the entire Wikipedia database into an instance of Apache Solr, an open-source search engine. Then querying words or pairs of words and reading the number of documents matching the query allowed us to get in constant-time[1] the few statistics needed to compute the topics coherence.

UCI topic coherence measure. For each of the four corpora [ap], [20news], [nips], and [nsf] we ran LDA with 30 topics. For each topic k, we isolated two sets of words: top ten words $S_k^{(\text{freq})}$ in frequency $p(w|k)$, and top ten $S_k^{(\text{rel})}$ words in relevance $p(w|k)e^{-H_w}$. We then compute improvement factor

$$\rho_k = \frac{\text{Coherence}_{\text{UCI}}\left(S_k^{(\text{rel})}\right)}{\text{Coherence}_{\text{UCI}}\left(S_k^{(\text{freq})}\right)}$$

in topic coherence from top words in frequency to top words in relevance. A factor ρ_k greater than one means that the top words in relevance are more coherent than the top words in frequency, according to the UCI metric. Histograms in Figure 3.2 shows the distribution of ρ_k for the four datasets. We can see that choosing top words based on relevance over top words based on frequency almost always increase the topic coherence.

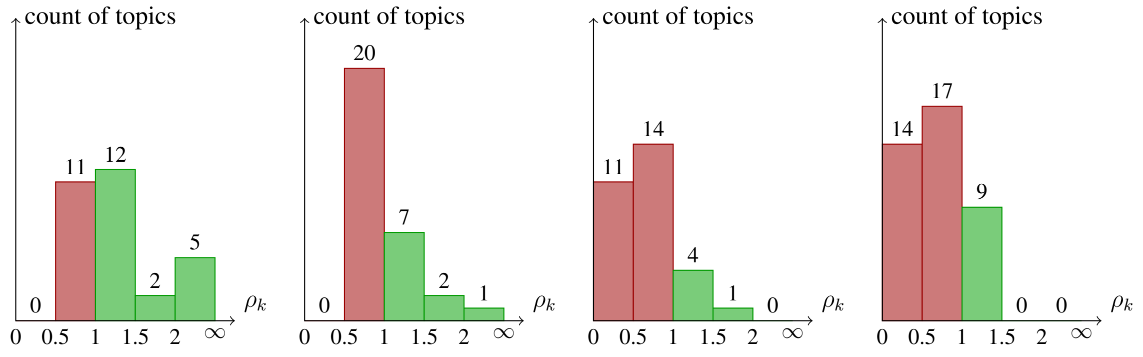


(a) Dataset [ap] (b) Dataset [20news] (c) Dataset [nips] (d) Dataset [nsf]

Figure 3.2: Distribution over 30 topics of the improvement factor ρ_k of topic UCI coherence from top words by frequency to top words by relevance for four datasets. Green bars indicate ranges where topic coherence is increased ($\rho_k > 1$).

[1]The time Solr takes to return the count may depend on the size of the index, and therefore the number of documents. For us, it was 3ms per request, small enough to consider it as constant-time.

UMass topic coherence measure. However, conducting the same experiment with the UMass measure does not give similar results. Changes in topic coherence becomes less predictable. Histograms in Figure 3.3 shows the distribution of the improvement factor according to this measure for the four datasets.



(a) Dataset [ap] (b) Dataset [20news] (c) Dataset [nips] (d) Dataset [nsf]

Figure 3.3: Distribution over 30 topics of the improvement factor ρ_k of topic UMass coherence from top words by frequency to top words by relevance for four datasets. Green bars are for $\rho_k > 1$.

Varying the number of topics. Let's now try to understand what is the cause of improvement in UCI measure. As sometimes, good results are due to the small number of topics used, we reconducted the same experiment with 100 topics instead of 30. Figure 3.4 presents results almost as good as for the first case.

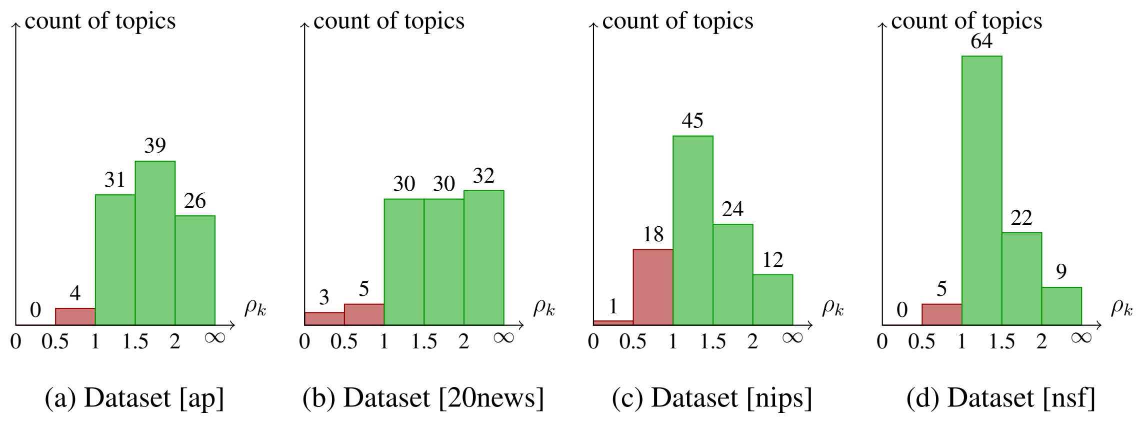


Figure 3.4: Distribution over 100 topics of the improvement factor ρ_k of topic UCI coherence from top words by frequency to top words by relevance for four datasets. Green bars are for $\rho_k > 1$.

Intrinsic PMI. For our last experiment, we consider an intrinsic version of the UCI measure. Instead of using Wikipedia as an external dataset, we used the same corpus on which LDA has been run to generate the word co-occurrence statistics. Figure 3.5 presents the results; computing the coherence score with this new intrinsic PMI-based measure, we a systematic improvement in topic coherence by choosing top words by relevance.

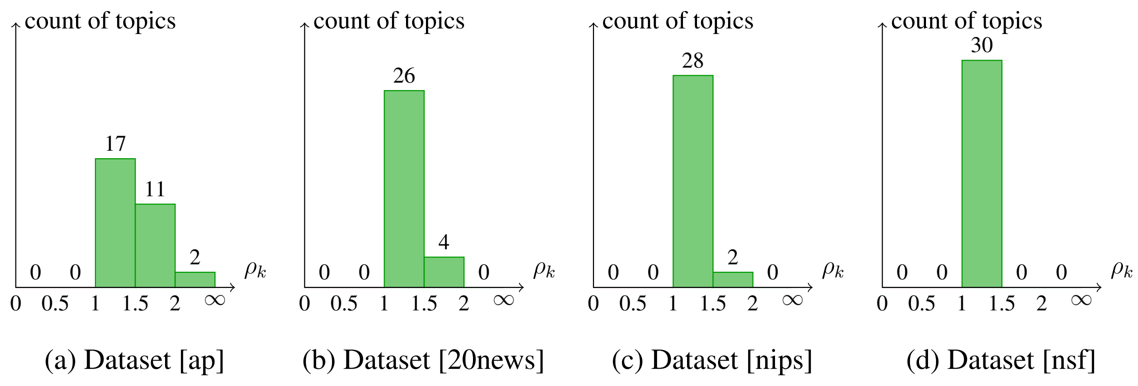


Figure 3.5: Distribution over 100 topics of the improvement factor ρ_k of topic coherence according to the intrinsic PMI-based coherence measure. Green bars are for $\rho_k > 1$.

The conclusion we can draw from these experiments is that we can almost systematically increase the average (intrinsic or extrinsic) PMI score

$$\sum_{i<j} \text{PMI}(w_i, w_j) \qquad \text{with} \quad \text{PMI}(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}$$

of the ten top words w_1, ..., w_{10} by choosing them by frequency to relevance.

3.6 Using PMI for topic relevance

Given the conclusion of the previous section, one can ask if it is always good thing to increase the PMI score. Indeed, the major issue with PMI is that it over-estimates low-frequency events. Therefore, a high PMI may not mean a high word correlation, but maybe just low-frequency words.

Example. For instance, the PMI is maximal when w_i and w_j always occur together:

$$D(w_i) = D(w_j) = D(w_i, w_j) = n$$

where n is the count of documents where their occur. Then their PMI will be

$$\text{PMI}(w_i, w_j) = \log \frac{\frac{n}{D}}{\frac{n}{D} \cdot \frac{n}{D}} = \log D - \log n$$

where D is the total number of documents. So for the same high predictive power of one word given the other, if they are present in all documents, their PMI will be zero, but it will be $\log D$ if they are appear in only one document.

Alternatives. Some alternatives have been developed to go around this issue. One is to use variants of the PMI such as the Weighted PMI (Schneider, 2005):

$$p(w_i, w_j) \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}$$

or giving more weight to the joint probability:

$$\log \frac{p(w_i, w_j)^2}{p(w_i)p(w_j)} \qquad \text{or} \qquad \log \frac{p(w_i, w_j)^3}{p(w_i)p(w_j)}$$

But the most common common alternative is to heuristically choose a threshold on frequencies and don't consider low-frequency events (Pantel and Lin, 2002).

PMI for top words. In our case, we are actually not computing the PMI for every pairs of words but just for the top ten words, either according to frequency or topic relevance. This is close to setting a threshold on frequency: we are only considering frequent words. This argument is true also for word relevance, even if we penalize some high-frequency words (the background words), a word scoring high on relevance $p(w|k)e^{-H_w}$ will have a high frequency $p(w|k)$ as well.

So by working only with high-frequency words, high PMI is more likely to mean high correlation rather than rarer events.

3.7 Conclusion

We introduce the *word relevance*, a novel measure $\mathcal{R}(w|k) \triangleq p(w|k)e^{-H_w}$ to order words within a topic, and therefore to represent the topic, by penalizing words that are shared in multiple topics. We found out that the top words by relevance are better representative for a topic, and confirmed that the average PMI is a good way of measuring topic coherence.

Chapter 4

Interactive LDA

This chapter introduces a novel interactive method for topic modeling, allowing the user to give guidance towards better topics. Work in the previous topic on words relevance within a topic is used to provide the user with better representations of topics, allowing him to give better guidance in the parameter search.

Running LDA in an interactive way allows the inference algorithm to use user feedback to direct the parameter search. The function that LDA is optimizing has multiple local optimum, and inference algorithms known most likely will not find the global optimum, but only a local one. Live user feedback is repeatedly used to kick the algorithm out of its local minimum, until it converges again to another one which may be better from the user perspective.

4.1 Previous work

The only approach that has been explored is to enforce pairwise word constraints in the prior over topics φ_k and re-run the algorithm after each user input.

Pairwise word constraints

Andrzejewski et al. (2009) replace the symmetric Dir(β) prior by a new one over topics to enforce some word constraints based on external sources: co-occurrences statistics, expert input, etc. It is only at the end of their work that they add a section on how to use this framework for interactive topic modeling: the external source is the live user instruction. Standard LDA is first run, discovering a latent topics for the corpus using Gibbs sampling. Then the four following steps are repeated until the user is satisfied with the topics:

1. The user gives feedback on the topics in three different forms: "those words must be in the same topic", "those words must not be in the same topic", and "those words should be isolated in one topic".
2. Each of these three feedback is encoded into two kinds of pairwise constraints: Must-Link(w_1, w_2) meaning that words w_1 and w_2 must be in the same topic, and Cannot-Link(w_1, w_2) meaning that they cannot be in the same topic.
3. A complex prior over words, based on Dirichlet Forests, is constructed to enforce those Must-Link and Cannot-Link constraints.
4. Gibbs sampling is started over using this new prior instead of the symmetric Dir(β).

A more efficient approach

This method requires to re-run Gibbs sampling from the beginning after each user instructions. Hu et al. (2011) extended this approach proposing Interactive Topic Modeling (ITM) where the Gibbs sampler is not restarted after each user action. Instead, the prior is updated in-place to incorporate the new constraints and the internal state of the Gibbs sampler is changed. This new state is then used as starting position for a new Markov chain.

The collapsed Gibbs sampler maintains the topic assignment z_{di} of each word w_{di}, along with some counts. Updating the internal state is done by state ablation; invalidate some topic-word assignments by setting $z = -1$. Because we are removing some word-topic assignments, the counts maintained by the Gibbs sampler are decremented accordingly:

count of words assigned to topic k in document d, and count of times word w is assigned to topic k in corpus.

They explore several strategies of invalidation: invalidate all assignments, only of documents that have any of the terms constraints, only of the terms concerned, or none. After each user actions, the Gibbs sample runs for 30 more iterations before asking for user feedback again.

Experiments were conducted using Amazon Mechanical Turk, having users providing the instructions on words. They used the dataset [20news], built a classifier on top of the final topic model, and measured one topic model performance in terms of classification error rate. However, given a corpus of documents, there can be several different coherent but orthogonal classifications that experts can agree on. The dataset might be organized according to one, and the algorithm discovering another, yet having a high classification error rate compared to the first one.

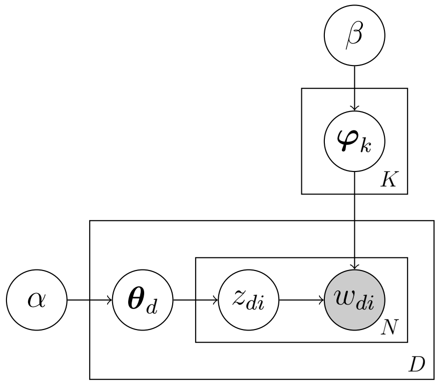
The results are not convincing; they only used one dataset, and the median user had an error reduction indistinguishable from zero.

The authors currently are preparing an extended version of the paper to be published in 2013 presenting a complete system for interactive topic modeling, with a web-based user interface.

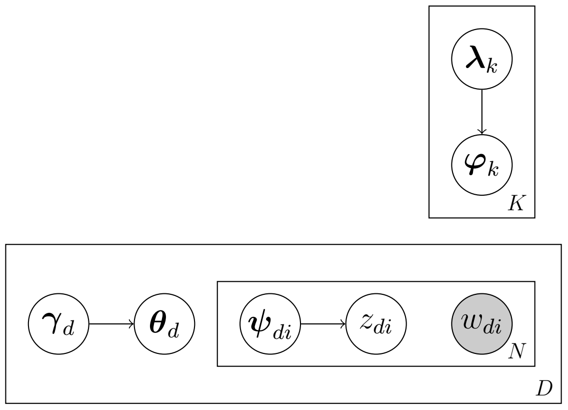
A novel method

In this chapter, we present a novel method for interactive topic modeling. Even though LDA is often introduced along with its Gibbs sampling inference algorithm because of its simplicity, our method is based on the other common inference approach for LDA, Variational Expectation-Maximization (EM) inference.

Our method allows us to be more expressive in terms of user feedback, where the previously described framework only allows pairwise Must-Link and Cannot-Link constraints. We can, for instance, specify that one topic is bad without wanting to enforce anything specific about the words used to describe that topic. The algorithm will redistribute them to other topics automatically.



(a) LDA model



(b) Variational approximation

Figure 4.1: Plate notation illustrating the changes made to approximate the posterior by Variational EM.

In our method, we don't build a complicated prior over words to enforce constraints. The resulting derivations are less complicated and intuitive.

4.2 Variational Expectation-Maximization

The goal of LDA inference is to compute the posterior $p(\boldsymbol{\Phi}, \boldsymbol{\Theta}|\boldsymbol{w})$ over the latent parameters $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ given the documents $\boldsymbol{w}$. This posterior distribution being computationally intractable, Variational EM approximates it using a tractable family of distributions. It is done be removing some dependencies between parameters.

Instead of having all topics $\boldsymbol{\varphi}_k$ share the same prior Dir(β), we have one different prior Dir$(\boldsymbol{\lambda}_k)$ for each topic k. We follow the same idea for documents: instead of having all documents sharing the same prior Dir(α), we have a different prior Dir$(\boldsymbol{\gamma}_d)$ for each document d

$$\forall k,\ \boldsymbol{\varphi}_k \sim \text{Dir}(\boldsymbol{\lambda}_k) \qquad \forall d,\ \boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\gamma}_d)$$

as illustrated in plate notation in Figure 4.1.

The variational EM algorithm for LDA is described in Algorithm 4 (Blei et al., 2003). First topics are initialized randomly, then EM epochs are repeated until convergence.

Algorithm 4 Variational EM for LDA

```
function VARIATIONALEM
    Initialize $\boldsymbol{\lambda}$ randomly
    repeat
        call EMEPOCH($\boldsymbol{\lambda}$)
    until convergence
```

In the Variational EM setup, we don't compute the topics $\boldsymbol{\varphi}_k$ and the document tagging $\boldsymbol{\theta}_d$ directly, instead we compute their distribution, namely the parameter $\boldsymbol{\lambda}_k$ of the Dirichlet prior over the topics $\boldsymbol{\varphi}_k$, and the parameter $\boldsymbol{\gamma}_d$ of the Dirichlet prior over the document tagging $\boldsymbol{\theta}_d$. Usually, we desire a value for the parameters $\boldsymbol{\varphi}_k \sim \text{Dir}(\boldsymbol{\lambda}_k)$ and $\boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\gamma}_d)$, therefore return the expected values

$$\mathbb{E}[\boldsymbol{\varphi}_k] = \frac{\boldsymbol{\lambda}_k}{\sum_w \lambda_{kw}} \quad \text{and} \quad \mathbb{E}[\boldsymbol{\theta}_d] = \frac{\boldsymbol{\gamma}_d}{\sum_k \gamma_{dk}}$$

which is just the normalized Dirichlet parameter. Because $\boldsymbol{\lambda}_k$ is very close to $\boldsymbol{\varphi}_k$ that we are going to output, we will be referring at $\boldsymbol{\lambda}_k$ as "the topic k".

The EM epoch procedure is presented in Algorithm 5. In the E step, documents are tagged with topics, keeping the topic $\boldsymbol{\lambda}_k$ fixed. The tagging is done by fitting the topics to the documents, computing for each word w_{di} a discrete distribution $\boldsymbol{\psi}_{di} \in \Delta_K$ over the K topics. In the M step, topics $\boldsymbol{\lambda}_k$ are updated according to the assignments ψ_{dwk} made in the E step.

Algorithm 5 One Variational EM epoch

```
function EMEPOCH($\boldsymbol{\lambda}$)
    for $d = 1$ to $D$ do                                          ▷ E step
        Initialize $\gamma_{dk} = 1$
        repeat
            Set $\psi_{dwk} \propto \exp \mathbb{E}[\log \theta_{dk} | \boldsymbol{\gamma}_d] \times \exp \mathbb{E}[\log \varphi_{kw} | \boldsymbol{\lambda}_k]$
            Set $\gamma_{dk} = \alpha + \sum_w \psi_{dwk} n_{dw}$
        until $\frac{1}{K} \sum_k |\text{change in } \gamma_{dk}| < 0.00001$

    Set $\lambda_{kw} = \beta + \sum_d n_{dw} \psi_{dwk}$                ▷ M step
```

Note that there are no assumptions on the topics $\boldsymbol{\lambda}$ at the beginning of an EM epoch. Even though it may delay the convergence, changing $\boldsymbol{\lambda}$ after the M step keeps the algorithm correct. We use that property to update the topics on input feedback in between epochs.

Note as well that document taggings $\boldsymbol{\gamma}_d$ are recomputed at each epoch, and only $\boldsymbol{\lambda}$ is kept between epochs. This motivates to work only on $\boldsymbol{\lambda}$.

4.3 Interpretation of Dirichlet parameters $\boldsymbol{\lambda}$

The parameters of a Dirichlet distribution have also an intuitive interpretation. Each component k represents the count of observations for event k. Therefore, λ_{kw} can be interpreted as the number of times word w was assigned to topic k throughout the corpus.

From the M step of the Algorithm 5, the sum of each column of $\boldsymbol{\lambda}$ without smoothing β is equal to the total number of occurrences $C(w) \triangleq \sum_d n_{dw}$ of word w in the corpus:

$$\sum_k (\lambda_{kw} - \beta) = \sum_k \sum_d n_{dw} \psi_{dwk} = C(w)$$

This gives us an interpretation of the Variational EM algorithm. It is splitting up the total count of occurrences $C(w)$ of each word w into the K topics, and adds smoothing β to get topics $\boldsymbol{\lambda}$. Then each row of $\boldsymbol{\lambda}$ is normalized to get outputted topic $\boldsymbol{\varphi}_k$.

Let's see an example in order to understand this intuition. Let's suppose our corpus is only made of the two following documents: "Pixel cafe" and "The pixel cafe is a weekly seminar". After removing the stop words, our vocabulary is composed of the four words: "pixel", "cafe", "weekly" and "seminar".

$$\boldsymbol{\lambda} = \begin{array}{|c|c|c|c|l}
\text{pixel} & \text{cafe} & \text{weekly} & \text{seminar} & \\
\hline
\beta + 1.8 & \beta + 2 & \beta + 0 & \beta + 0.1 & \leftarrow \boldsymbol{\lambda}_1 \\
\hline
\beta + 0.2 & \beta + 0 & \beta + 1 & \beta + 0.9 & \leftarrow \boldsymbol{\lambda}_2 \\
\hline
\end{array}$$

This matrix shows what the Variational EM algorithm could have discovered in this corpus of two documents with $K = 2$ topics. Words "pixel" and "cafe" occur twice in the corpus, and words “weekly” and “seminar” occur only once. These counts can be found in $\boldsymbol{\lambda}$ by summing each columns without the smoothing. Note that counts don’t have to be integers.

Moreover, smoothing parameter β is often set to $\frac{1}{K}$. So adding β to every element of column w is equivalent of having one extra occurrence of word w split equally across all K topics.

4.4 Types of user feedback

We can implement all sorts of actions on the topic matrix $\boldsymbol{\lambda}$. In this work, we restricted ourselves to four types of feedback. Each of the following feedback has a dedicated section describing it in details.

“These words do not belong to this topic.” A natural feedback a user can give about a topic is that some of the words presented do not belong to it. Chang et al. (2009) measured quality of topics by asking user to find a random word they intruded into the topic’s top words. We will respond to this feedback by resetting the count of these words in topic k, which will have the effect of removing them from topic k, and assigning them to some other topics.

“This topic does not make any sense.” When there is really no logic tying some words together in the topic, we will simply remove the corresponding row in $\boldsymbol{\lambda}$, which will have the effect of removing the topic. Words from that topics will naturally be assigned to the closest other topic in the next EM epoch.

“These two topics are duplicates.” When two topics are so similar that one cannot differentiate one from another, we merge the two topics by simply adding the counts of the two topics. This will have the effect of having one topic as heavy as both previous combined and describing the two.

"This topic is a mixture of two different topics." When the user can see in the same topic two groups of words, each tied by a different logic, we create a new row in the matrix and split the word counts between the original row and the new one, according to the two groups of words the user gave. This will have the effect of splitting the topic into two different topics.

4.5 Removing words from topics

We remove word w from topic k just by resetting its count in the topic $\boldsymbol{\lambda}_k$ (Algorithm 6). Note that we keep the smoothing parameter β to prevent taking the logarithm of zero in the E step.

Algorithm 6 Word removing from topics

```
function REMOVE(word $w$, topic $k$)
    Set $\lambda_{kw} = \beta$
```

Let's now try to understand the effect of that update on the algorithm. Variational topic assignment ψ_{dwk} of each word w in each document d is done in the E step:

$$\psi_{dwk} \propto \exp \mathbb{E}[\log \theta_{dk} | \boldsymbol{\gamma}_d] \times \exp \mathbb{E}[\log \varphi_{kw} | \boldsymbol{\lambda}_k].$$

Note that this is different from the topic assignment z_{di}. Assignment $z_{di} \in [\![1, K]\!]$ holds the topic that generated the word at position i in document d, whatever this word is. On the other hand, $\boldsymbol{\psi}_{dw} \in \Delta_K$ describes the distribution over the K topics of every word w of the vocabulary for document d, whether this word is observed in d or not.

The expression in the update rule for ψ_{dwk} is an increasing function in λ_{kw} and γ_{dk}. It means that the higher λ_{kw}, i.e. the more occurrences of word w were assigned to topic k, the more topic k is going to have responsibility in observing the word w in d. Also, the higher γ_{dk}, i.e. the more words from document d were assigned to topic k, the more topic k is going to have responsibility in observing the word w in d. Table 4.1 presents these two factors pushing a word into a topic, the first concerning topic-word bonds, and the second document-topic bonds.

Table 4.1: Factors controlling topic assignment ψ_{dwk} of word w in document d

Factor	Controlled by
Topic-word bonds How much word w is already part of topic k	**Pseudocount** λ_{kw} How many occurrences of word w were assigned to topic k
Document-topic bonds How much topic k is present in document d	**Pseudocount** γ_{dk} How many words of documents d were assigned to topic k

Resetting pseudocount λ_{kw} to β will kill the topic-word factor, but the word may still return to the topic if the other factor is strong: if the word is part of documents that have a high proportion of topic k (γ_{dk} big), then it might return back to this topic (λ_{kw} might increase).

As we reset a value in the column w of $\boldsymbol{\lambda}$, the column without smoothing does not add up to the number total number $C(w)$ of occurrences of w anymore:

$$\sum_k (\lambda_{kw} - \beta) < C(w)$$

This property not holding is not a problem, as it was not a constraint, but a consequence of the update rules. We simply lose the deleted occurrences from our prior belief. At the end of the next EM epoch, all occurrence of w will be assigned to topics and this property will be restored.

Experiments

Table 4.2 shows the top words ranked by relevance of a topic about cryptography. Suppose the user gives the feedback that words *israel* and *lebanese* do not belong to that topic, so we call the removing procedure on those two words, then we run EM epochs until convergence. The second columns shows the new top words after the operation. The words that took the place of the removed words are *phones* and *crypto* that belong well to the cryptography topic. On this example, removing two words from a topic produced a more coherent topic. Note that this this experiment and all following, words presented to describe a topic are top words by relevance $p(w|k)e^{-H_w}$.

Table 4.2: Top words of one topic discovered by running LDA on corpus [20news] with 30 topics, before and after removing words *israel* and *lebanese*, and running LDA until convergence.

Before	After
clipper	*clipper*
escrow	*escrow*
encryption	*encryption*
chip	*chip*
keys	*keys*
sternlight	*sternlight*
encrypted	*encrypted*
israel	*wiretap*
lebanese	*phones*
wiretap	*crypto*

But we have seen that when the document-topic bonds are strong, words may come back into the topic. To get a sense of how much this could be a problem, we made the following experiment. First, we ran LDA until convergence, discovering 30 topics. Then, for each of the top ten words of each topic, we tried to remove it, run LDA until convergence, and compute how much the word came back to the topic it was deleted from by computing the following *come-back ratio*:

$$\text{ratio} = \frac{\text{count}(\text{occurrences of } w \text{ assigned to topic } k \text{ before})}{\text{count}(\text{occurrences of } w \text{ assigned to topic } k \text{ after})} = \frac{\lambda_{kw}^{(\text{before})} - \beta}{\lambda_{kw}^{(\text{after})} - \beta}.$$

Figure 4.2 shows the distribution of this ratio over the 300 words (10 words times 30 topics). The are two clear categories of words: those that don't come back at all in the topic (73%), in green, and those that come back completely, in red, having a ration close to one, and even larger sometimes.

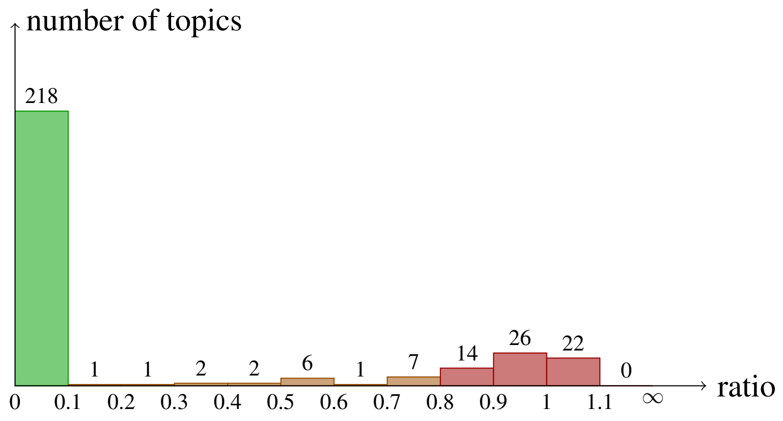


Figure 4.2: Distribution of the fraction of the count of the deleted word that comes back to the topic after removal.

To inquire what was making a word to come back or not to a topic after being deleted, we looked at the number of documents containing each words. Figure 4.3 presents the distribution of the number of document that contains the deleted word for the two groups: words that did not come back to the topic (ratio lower than 0.5, in green), and words that did come back (ratio greater than 0.5, in red). We can see that words that come back to the topic they were deleted from, are the ones that are present in very few documents (maximum four). Because each of these words w was a strong indicator of the topic k it was deleted from, we know those very few documents are strongly tied to topic k, which makes γ_{dk} high enough to increase back λ_{kw}.

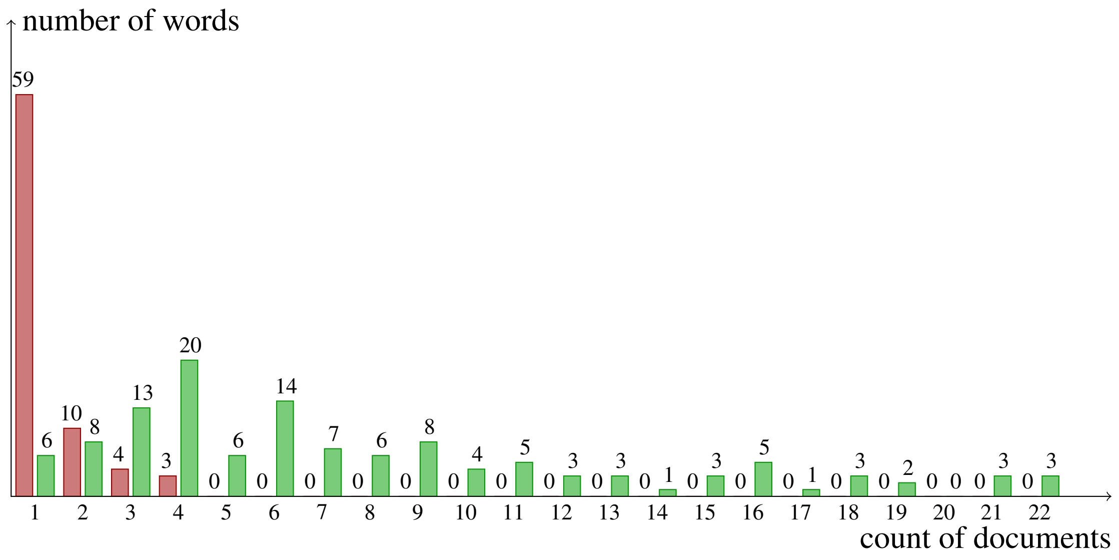


Figure 4.3: Distribution of the number of documents containing the removed word for words that did (green) and did not (red) come back. The green series keeps going after 22, but what is important to notice is that the red stops at a count of four documents.

We tried another strategy to remove words: we not only reset the count for the given word w, but we decrease the count of every other word w' proportionally to how much w trigers w', measured by $p(w'|w)$. We reconduct the same experiment as before with this new strategy. Figure 4.4 shows the distribution of the come-back ratio over the 300 words. The number of words that do not come back to the topic from which they were removed stays the same (73%). The only difference is that the second group in the histogram is slightly translated to the left, so words that do come back, tend to come back in smaller proportion.

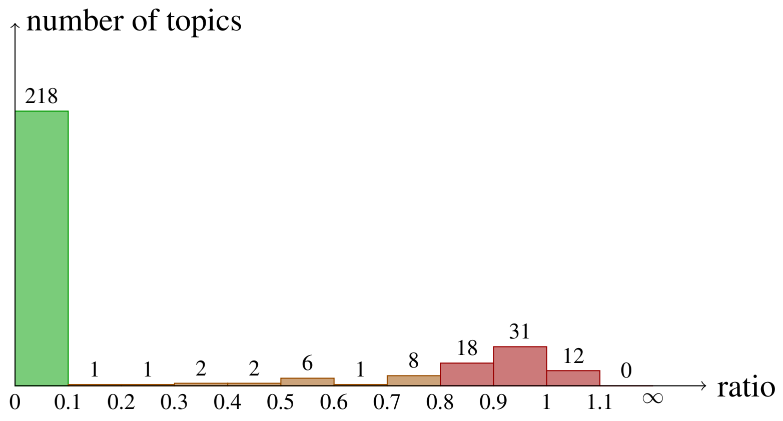


Figure 4.4: Distribution of the fraction of the count of the deleted word that comes back to the topic after removal with the new strategy.

4.6 Deleting topics

We implement deletion of topics by simply removing the corresponding row in the topic matrix $\boldsymbol{\lambda}$ and decrementing the number of topics K by one. Algorithm 7 presents the deletion procedure. Because the document tagging matrix $\boldsymbol{\gamma}$ is recomputed at each epoch, we don't need to worry about removing a column in it, it will be built with the correct dimensions in the next EM epoch.

Algorithm 7 Deleting topics

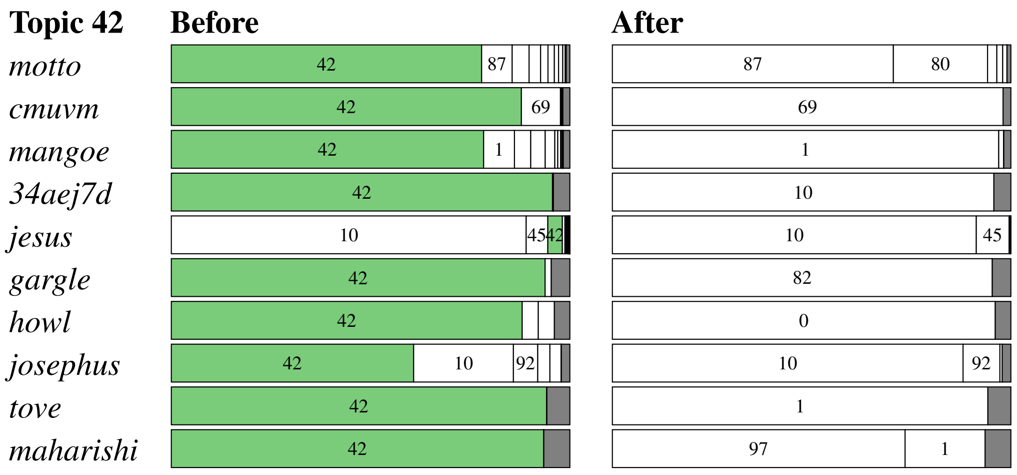
```
function DeleteTopic(topic $k$)
    Remove row $k$ in $\boldsymbol{\lambda}$
    Set $K = K - 1$
```

Here as well, as we deleted a row in the topic matrix $\boldsymbol{\lambda}$, columns without smoothing don't add up to the word occurrences anymore. But at the next epoch, all the words of the corpus are going to be assigned to the remaining topics, and this property will be restored.

Experiments

In this experiment, we show the effect of deleting a topic on each of its words. The first column of Table 4.3 presents the top words of an example of non-coherent topic, where words are not related to each other. Next to each word, we display the discrete distribution over topics, before and after deleting the topic. The topic to be deleted is represented in green. We can see that the words are usually caught by the largest other topic than the deleted one. For example, word *josephus* will be caught by topics 10 and 92, that was already present in the distribution of *josephus* before deleting topic 42.

Table 4.3: Distribution over topics of the top words of the topic to be deleted, before and after deleting it. Experiment done after running LDA on the corpus [20news] with 30 topics until convergence.



4.7 Merging topics

When we want to merge two topics k_1 and k_2, we want a new topic that accounts for all words generated by k_1 and k_2. We have seen that the parameters $\boldsymbol{\lambda}_k$ of Dirichlet distribution can be interpreted as pseudocounts for the discrete distribution $\boldsymbol{\varphi}_k \sim \text{Dir}(\boldsymbol{\lambda}_k)$. If we have two sets of pseudocounts $\boldsymbol{\lambda}_{k_1}$ and $\boldsymbol{\lambda}_{k_2}$, adding them together will be as if all pseudocounts were accounting for the same topic. We just need to remove the smoothing β once as it is in both topics.

Algorithm 8 presents the merging procedure. Given topic k_1 and k_2, the topic k_1 will keep the distribution $\mathrm{Dir}(\boldsymbol{\lambda}_{k_1} + \boldsymbol{\lambda}_{k_2} - \beta)$ in slot k_1 and the topic k_2 will be deleted.

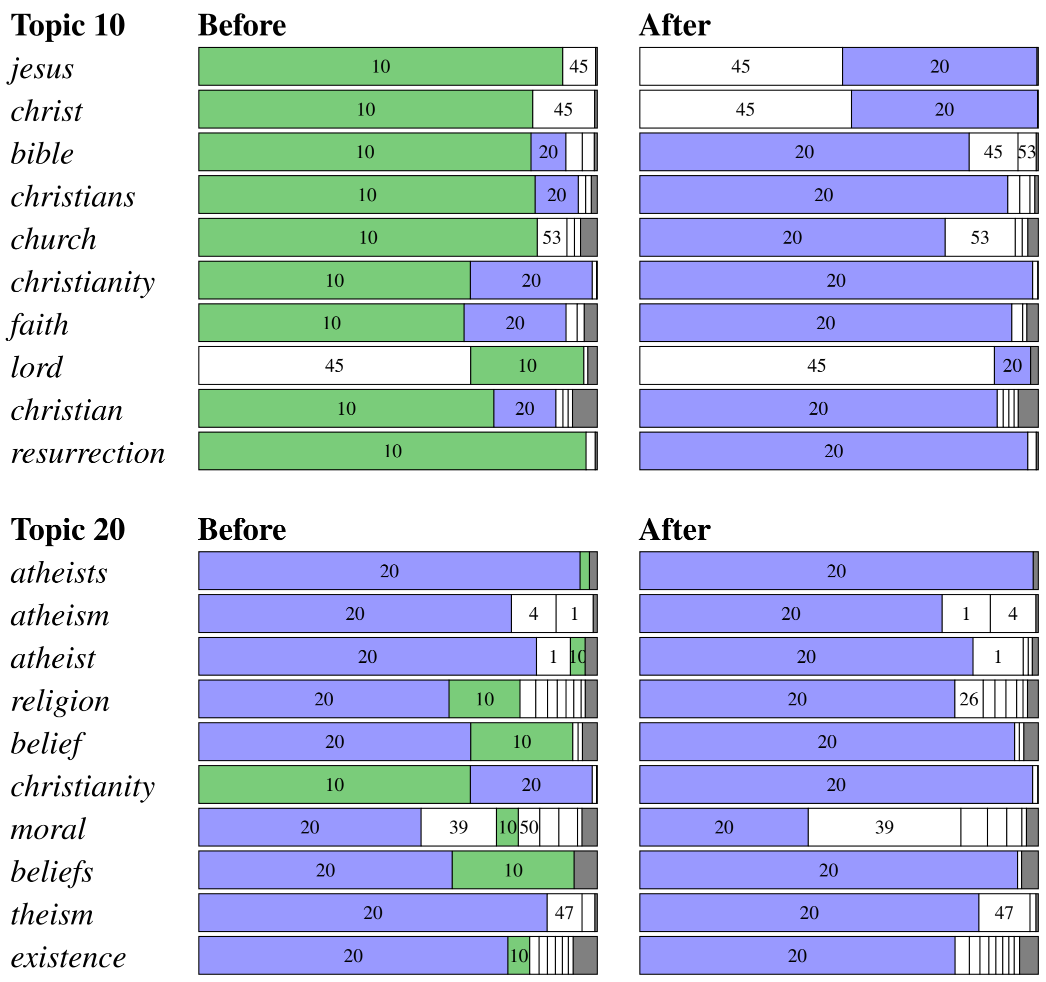
Algorithm 8 Merging topics

```
function MERGETOPICS(topic $k_1$, topic $k_2$)
    Set $\lambda_{k_1 w} = \lambda_{k_1 w} + \lambda_{k_2 w} - \beta$
    call DELETETOPIC($k_2$)
```

Experiment

This experiment shows the effect of merging two topics on the top words of each. Table 4.4 presents the top ten words of first topic 10 (in green), and then topic 20 (in blue), both about religion, that we are going to merge. Next to each words, its discrete distribution over topics before and after the merge.

Table 4.4: Distribution over topics of the top words of the two topic to be merged, before and after the merging. Experiment done after running LDA on the corpus [20news] with 30 topics until convergence.



We merged topics 10 and 20 into topic 20, deleted topic 10, and run LDA until convergence. We can see in the third column that after the merge, all the top words of both former topics 10 and 20 are now caught by the same topic 20. We have indeed built a new topic capturing both former topics 10 and 20.

4.8 Splitting topics

When merging two topics, we only need to know which topics are to be merged. To split one topic k_1 into two, we need more information; the user has to provide two sets of words W_1 and W_2 as a seed to initiate the split.

After a split, we will get one more topic. We add a row k_2 in the matrix $\boldsymbol{\lambda}$, initialize it to β, and for each word w, we are going to move a fraction

$$\rho_w \triangleq \frac{p(w|W_2)}{p(w|W_1) + p(w|W_2)}$$

of the count from row k_1 to row k_2. This fraction is zero for words in W_1 and one for words in W_2; words of W_i will be completely in topic k_i. For each of the remaining words w, we are going to move the fraction ρ_w of counts depending on how close w is from W_2 compared to W_1. Algorithm 9 presents the splitting procedure.

Algorithm 9 Splitting topics

```
function SPLITTOPIC(topic $k_1$, set of words $W_1$, set of words $W_2$)
    Add a row $k_2$ in matrix $\boldsymbol{\lambda}$
    Set $\rho_w \triangleq \frac{p(w|W_2)}{p(w|W_1)+p(w|W_2)}$
    Set $\lambda_{k_2 w} = \rho_w \, \lambda_{k_1 w}$
    Set $\lambda_{k_1 w} = (1 - \rho_w) \, \lambda_{k_1 w}$
    Set $K = K + 1$
```

Now computing the probability of a word v_i, given some other words v_0, ..., v_{i-1}, is hard as it requires to compute the join

$$\begin{aligned} p(v_0, ..., v_{i-1}, v_i) &= \sum_{z_0,...,z_i,d} p(d) \prod_{j=0}^{i} p(v_j|z_j)\, p(z_j|d) \\ &= \sum_{z_0,...,z_i,d} \frac{N_d}{D} \prod_{j=0}^{i} \varphi_{z_j,v_j}\, \theta_{d,z_j} \end{aligned}$$

This computation requires $O(KDW^{12})$ floating-point operations, so is intractable. Another approach would to compute this joint empirically, but we can expect for most the the word combination v_0, ..., v_i that we consider, there will be no document with all those words, and so $p(v_0, ..., v_{i-1}, v_i) = 0$.

So we tackle the problem differently. Computing $p(w|W_1)$ means computing the distribution over the vocabulary inferred by the words of W_1. This is actually the same thing we are doing for each document in the inference algorithm: approximate the distribution that a document infers over words by a linear combination of topics. So we use the E step of the inference algorithm, treating W_1 as a document. It will give us a distribution over topics $p(k|W_1)$, and therefore a distribution over words:

$$p(w|W_1) = \sum_k p(w|k)\, p(k|W_1)$$

This allows us to compute $p(w|W_i)$, and therefore the fraction ρ_w for each word to do the split.

Experiments

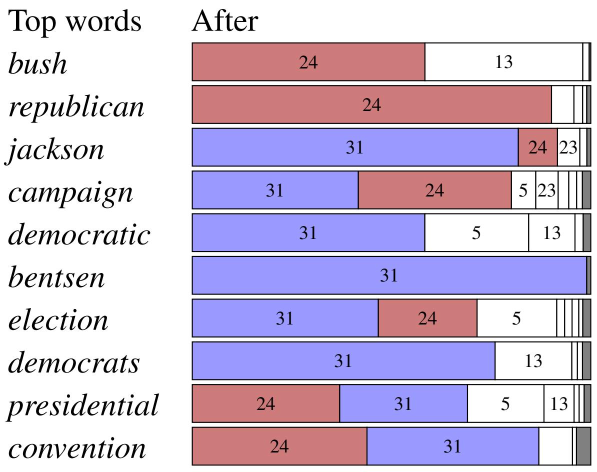
In this experiment, we show an example of splitting a topic. We first ran LDA on the corpus [ap] with 30 topics until convergence. Then, we split the topic that we labeled "Politics" in Table 4.5 using the two seed words *republican* and *bush* for the first topic and the three words *democrats*, *bentsen*, and *democratic* for the other one. Running LDA again until convergence successfully discover two stable topics, one about republicans, the other about democrats.

Table 4.5: Topic "Politics" has been split into two topics using seed words *republican* and *bush* for the first, and *democrats*, *bentsen*, and *democratic* for the second. LDA was then run until convergence.

Before	After	
"Politics"	"Republicans"	"Democrats"
bush	*bush*	*jackson*
republican	*republican*	*democratic*
jackson	*campaign*	*bentsen*
campaign	*presidential*	*democrats*
democratic	*dole*	*campaign*
bentsen	*robertson*	*election*
election	*convention*	*vote*
democrats	*republicans*	*presidential*
presidential	*poll*	*governor*
convention	*election*	*votes*

Table 4.6 shows where the top words of the topic that have been split have went after the split. General words such as *campaign*, *election*, *presidential* and *convention* are almost equally shared between topics “Republicans” and “Democrats”, whereas words specific to one party is almost completely caught by the corresponding topic.

Table 4.6: Distribution over topics of the top words of the two topic that have been split, after the split. Experiment done after running LDA on the corpus [20news] with 30 topics until convergence.



4.9 Experimental Design

This section describes experiments that can be conducted at a larger scale. This first experiment is designed to measure the effectiveness of just using the actions deleting topics and removing words from topics. It is designed for a crowd-sourcing platform such as Amazon Mechanical Turk (AMT) which is often used to perform large-scale experiments (Hu et al., 2011; Chang et al., 2009; Crump et al., 2013).

We run LDA twice, first with 30 then 100 topics, on the four corpus [20news], [nips], [ap], and [nsf]. That gives us 520 topics to evaluate. To evaluate a topic, we add a random word to the set of the five top word and we ask users to find the intruder. If they can find it every time, then we can conclude the topic is coherent. Otherwise, we conclude that the topic is not coherent. Given a incoherent topic, if the distribution of the choices of intruder by users is close to uniform, then we delete the topic. Otherwise, we delete the words mainly

Table 4.7: The number of times a word has been chosen as an intruder by AMT users.

Word	Times
jehovah	3
elohim	1
cramer	2
henry	2
radar	2
solar	0

designated as intruder.

Here is a toy test we did on AMT with only one topic. The top words for the topic were

jehovah, elohim, henry, radar, solar, orbit, pluto, moon, atmosphere, lunar

Picking the five first and adding a random word (here, *cramer*), the intruder task instance was

jehovah, elohim, cramer, henry, radar, solar

Table 4.7 presents the result of the experiment, we asked ten different users to do the task and we can see the distribution is quite spread out and there is no clear intruder designated. So we would conclude the topic is incoherent and delete it at the next round.

After getting crowd feedback and updating the model, we run LDA again until convergence. Those two steps are repeated a couple of times. We measure the quality of a topic by the miss rate of the intruder task. Miss rate should decrease with the number of iterations.

Incentive

When designing a survey with multiple choice questions, we have to make sure users don't answer randomly. On AMT, we can reject answers that don't hit the intruder, but not only the user will not be paid, but also it will penalize his validation rate on AMT. This is not fair as for very bad topics, the user has no way to know which word is the intruder.

Another option is to give financial incentive to find the correct intruder. However, Crump et al. (2013) have studied task performance of AMT users when varying the amount of the financial incentive, either \$2 and a bonus up to \$2.5 based on task performance, or \$0.75 with no bonus. Results show that the amount on the incentive does not effect the task performance but does effect the rate at which workers sign up for the task.

Another idea, is to embed this experiment in a captcha of a large website. Then people will have real incentive to give the correct answer, just because it is getting into their way to subscribe. Generalizing this, having to answer correctly an intruder question to access a resource online will guarantee that users give an honest try to find the intruder.

4.10 Conclusion

In this section, we presented a novel way for interactive topic modeling, repeatedly using the user feedback to kick the parameter search out of a local optimum, yielding to better topics for the user's eyes. By changing the internal state of the Variational EM algorithm between epochs, we could express the most common user feedback in an intuitive manner.

Bibliography

AlSumait, L., D. Barbará, J. Gentle, and C. Domeniconi (2009). Topic significance ranking of lda generative models. In *ECML*.

Andrzejewski, D., X. Zhu, and M. Craven (2009). Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *ICML*, pp. 25–32.

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

Blei, D. M. (2012, April). Probabilistic topic models. *Commun. ACM 55*(4), 77–84.

Blei, D. M., A. Y. Ng, and M. I. Jordan (2002). Latent dirichlet allocation. In *NIPS*.

Blei, D. M., A. Y. Ng, and M. I. Jordan (2003, March). Latent dirichlet allocation. *J. Mach. Learn. Res. 3*, 993–1022.

Campbell, L. L. (1966). Exponential entropy as a measure of extent of a distribution. In *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, Volume 5.

Chang, J., J. Boyd-Graber, C. Wang, S. Gerrish, and D. M. Blei (2009). Reading tea leaves: How humans interpret topic models. In *NIPS*.

Chuang, J., C. D. Manning, and J. Heer (2012). Termite: Visualization techniques for assessing textual topic models. In *Advanced Visual Interfaces*.

Crump, M. J. C., J. V. McDonnell, and T. M. Gureckis (2013, March). Evaluating Amazon's Mechanical Turk as a Tool for Experimental Behavioral Research. *PLoS ONE 8*(3), e57410+.

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*(6), 391–407.

Ding, C., T. Li, and W. Peng (2008). On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics and Data Analysis 52*, 3913–3927.

Geman, S. and D. Geman (1984, November). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell. 6*(6), 721–741.

Griffiths, T. and M. Steyvers (2002a). A probabilistic approach to semantic representation. In *Proceedings of the 24th Annual Conference of the Cognitive Science Society*.

Griffiths, T. L. and M. Steyvers (2002b). Prediction and semantic association. In *NIPS*, pp. 11–18.

Hofmann, T. (1999). Probilistic latent semantic analysis. In *UAI*.

Hu, Y., J. Boyd-Graber, and B. Satinoff (2011). Interactive topic modeling. In *Association for Computational Linguistics*.

Lee, D. D. and H. S. Seung (1999). Learning the parts of objects by non-negative matrix factorization. *Nature 401*(6755).

Mimno, D., H. Wallach, E. Talley, M. Leenders, and A. McCallum (2011). Optimizing semantic coherence in topic models. In *EMNLP*.

Newman, D., Y. Noh, E. Talley, S. Karimi, and T. Baldwin (2010). Evaluating topic models for digital libraries. In *Proceedings of the 10th annual joint conference on Digital libraries*, New York, NY, USA, pp. 215–224. ACM.

Pantel, P. and D. Lin (2002). Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, New York, NY, USA, pp. 613–619. ACM.

Pauca, V. P., F. Shahnaz, M. W. Berry, and R. J. Plemmons (2004). Text mining using non-negative matrix factorizations. In *SDM*.

Schneider, K.-M. (2005). Weighted average pointwise mutual information for feature selection in text categorization. In *Proceedings of the 9th European conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD'05, Berlin, Heidelberg, pp. 252–263. Springer-Verlag.

Stevens, K., P. Kegelmeyer, D. Andrzejewski, and D. Buttler (2012). Exploring topic coherence over many models and many topics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, Stroudsburg, PA, USA, pp. 952–961. Association for Computational Linguistics.

Steyvers, M. and T. Griffiths (2006). Probabilistic topic models. In T. Landauer, D. Mcnamara, S. Dennis, and W. Kintsch (Eds.), *Latent Semantic Analysis: A Road to Meaning.* Laurence Erlbaum.

Tzikas, D., A. Likas, and N. Galatsanos (2008, November). The variational approximation for Bayesian inference. *IEEE Signal Processing Magazine 25*(6), 131–146.

Wallach, H. M., I. Murray, R. Salakhutdinov, and D. Mimno (2009). Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, New York, NY, USA, pp. 1105–1112. ACM.

Wei, X. and B. Croft (2006). Lda-based document models for ad-hoc retrieval. In *SIGIR*.