Ausarbeitungen für das Seminar

Algorithmic Methods in the Humanities – Summer 2016

1 Latent Dirichlet Allocation

Florian Becker

Abstract

Due to the vast amounts of texts which are produced and digitized on a daily basis, it becomes more and more difficult to find what we are looking for. Topic models can be used as a tool to discover *topics* in large collections of texts. Each text is considered to be a distribution over topics and topics are, in turn, distributions over words. Topic model algorithms receive unlabeled texts as input and produce the topic distribution as output. Latent Dirichlet Allocation is a generative probabilistic topic model. The basic idea is that documents are mixtures of latent topics distributed according to a Dirichlet prior.

1.1 Introduction

When seeking digitized and hyper-linked information, we use searching tools that weigh each result according to some centrality measure. Such techniques are often sufficient, provided that we only want to find some result and do not care about the underlying topic structure which might give us more insight into a document or even a collection of many documents. Topic models enable us to not only search documents by the themes that they contain, but also give us a tool with which we can automatically organize and structure text corpora. It is assumed that a document exhibits multiple topics and each is a distribution over words. Consider, for instance, a corpus with texts about digital humanities. A topic model algorithm might output that the text corpus contains the topics *Computer Science*, *Linguistics* and *Philosophy*. But since every topic is a distribution over words, the algorithm will not just output '*Computer Science*', but will rather give a distribution over words, which we then can identify as the topic '*Computer Science*': e.g. *algorithm, program, complexity, machine, Turing, artificial intelligence*, etc.
Topic Models can also be thought of as a way of clustering similar documents. Scientific papers, books, tweets, blogs etc. can be organized according to the themes they feature. With topic models organizing a large corpus becomes feasible.
Latent Dirichlet Allocation (LDA) is a generative topic model. This means that the topic and word distribution of a document can be explained by a generative model. In machine learning generative models are used to *simulate* data points. The goal is to infer the hidden parameters which are assumed to be known in a generative model. Hence, inference can be seen as reversing the generative process. In the case of Latent Dirichlet Allocation the hidden variables are the per-document topic distribution, the per-document per-word topic assignments and the topics themselves.

Latent Dirichlet Allocation was introduced by David Blei, Andrew Ng, and Michael Jordan in 2003 [2]. Since then it has been applied in various different domains to many different sorts of texts. It is not restricted to operate on 'classical' documents. It was applied, for instance, in the domain of public health, where [5] analyzed tweets with LDA in order to track illnesses over time. It was also just recently applied in bioinformatics. [6] annotated

genomes with a feature term that describes a feature with LDA.
As the intersection of computing and humanities grows, topic models will become more important. Humanities profit from algorithmic methodologies applied on large text corpora for obvious reasons; manually structuring those is mostly not an option. Moreover, one may gain insight about the semantic levels when examining the low-dimensional structure of a text corpus.
After a section about preliminaries and notation this report will discuss the generative model, which can also be seen as a concise way to formulate the conditions of Latent Dirichlet Allocation. Afterwards, the problem of Bayesian inference is discussed and a solution, the Gibbs Sampling algorithm, is presented.

1.2 Preliminaries and Notation

Throughout this report the notation of the original paper by Blei et al. will be used and is summarized in the following table.

Table 1 Notation, as used in [2].

symbol	**description**
α	parameter of the Dirichlet prior on the per-document topic distributions
β	parameter of the Dirichlet prior on the per-topic word distribution
θ_i	topic distribution for document i
φ_k	word distribution for topic k
z_{ij}	topic for the j-th word in document i, and
w_{ij}	specific word.

In order to understand in what way topics are assumed to be distributed the multinomial and Dirichlet distribution including important properties will be summarized briefly.

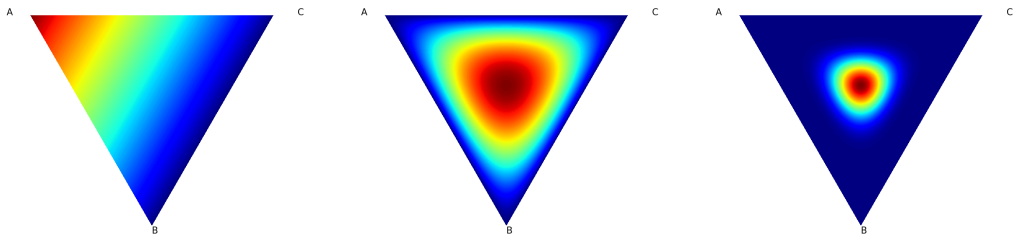
1.2.1 Multinomial Distribution

The binomial distribution is a discrete distribution over events with two outcomes. It gives the probability of obtaining k successes out of n trials, where each success has a probability of p. The probability mass function (PMF) is given by (Equation 1).

$$\Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{1}$$

The binomial distribution is a special case of the multinomial distribution. The multinomial distribution models events which have k outcomes, where each has a fixed probability. The probability mass function is given by

$$f(x_1, \ldots, x_k; n, p_1, \ldots, p_k) = \frac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k} \tag{2}$$



(a) $\mathrm{Dir}(\alpha = \langle 2, 1, 1\rangle)$ **(b)** $\mathrm{Dir}(\alpha = \langle 2, 2, 2\rangle)$ **(c)** $\mathrm{Dir}(\alpha = \langle 10, 10, 10\rangle)$

Figure 1 Dirichlet distributions in the two-dimensional simplex with different α. The corners A, B, C correspond to topics. Regions of higher probability are shown by a heat map, i.e. (dark) red corresponds to the most probable sample.

1.2.2 Dirichlet Distribution

The Dirichlet distribution $\mathrm{Dir}(\alpha)$ is a probability distribution parameterized by a positive real valued vector α. The probability density function is given by:

$$f\left(x_1, \cdots, x_K; \alpha_1, \cdots, \alpha_K\right) = \frac{\Gamma\left(\sum_{i=1}^K \alpha_i\right)}{\prod_{i=1}^K \Gamma(\alpha_i)} \prod_{i=1}^K x_i^{\alpha_i - 1} \tag{3}$$

where $\Gamma()$ is the Gamma function.

The Dirichlet distribution is often also expressed with the inverse of the Beta function as a normalizing constant. $\mathrm{B}(\boldsymbol{\alpha})$ is constant as it only depends on the fixed parameter $\boldsymbol{\alpha}$.

$$f\left(x_1, \cdots, x_K; \alpha_1, \cdots, \alpha_K\right) = \frac{1}{\mathrm{B}(\boldsymbol{\alpha})} \prod_{i=1}^K x_i^{\alpha_i - 1} \qquad \boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_K) \tag{4}$$

where

$$\mathrm{B}(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^K \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^K \alpha_i\right)}, \qquad \boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_K) \tag{5}$$

The expected value of the Dirichlet distribution is simply computed by:

$$\mathrm{E}[X_i] = \frac{\alpha_i}{\alpha_0} \tag{6}$$

where

$$\alpha_0 = \sum_{i=1}^K \alpha_i. \tag{7}$$

The support of the Dirichlet distribution of order K is the $(K-1)-$simplex. For $K = 3$ the $2-$simplex is given by a regular triangle, where each corner is an event. The vertices are $(0, 0, 1), (0, 1, 0)$ and $(1, 0, 0)$. A point $\vec{p} = (p_1, p_2, p_3)$ on this probability simplex corresponds to a certain event.

1.3 Plate Notation

In a probabilistic graphical model conditional dependencies between various random variables are often depicted in plate notation. Plate notation is a concise method for visualizing the factorization of the joint probability. Typically, shaded nodes stand for *observed* random variables. The plates represent the repeated structure.

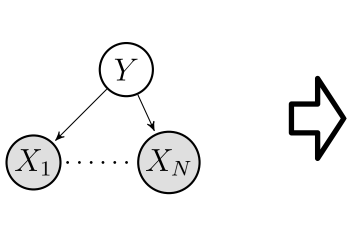


Figure 2 Plate notation as a way to summarize conditional dependencies. Shaded nodes represent *observed* variables. Here, the X_i are conditionally dependent on Y. In plate notation all the X_i are summarized as one plate.

The joint probability of the probabilistic graphical model depicted in (Figure 2) is

$$\begin{aligned} Pr[X_1, \dots, X_n] &= \prod_{i=1}^{n} Pr[X_i | \operatorname{parent}(X_i)] \\ &= \prod_{i=1}^{n} Pr[X_i | Y] \end{aligned} \tag{8}$$

The plate notation will be used in the next section to have a concise representation of Latent Dirichlet Allocation and especially the conditional dependencies.

1.4 Generative Process

Generative models generate data values according to a probability distribution. In machine learning generative models are used to estimate the joint probability distribution $Pr(X|Y)$ of observed data X and labels Y. In the context of probabilistic topic models, Y corresponds to the latent variables and X to the (observable) words in a corpus. Latent Dirichlet Allocation assumes the following generative process for a corpus:

```
Input : Number of words $N$, Number of topics $K$
        Dirichlet parameters $\alpha \in \mathbb{R}_+^K, \beta \in \mathbb{R}_+^N$
Output : $D$ documents (bag-of-words) with $N$ words each
1. Choose $\theta_i \sim \mathrm{Dir}(\alpha)$,
2. Choose $\varphi_k \sim \mathrm{Dir}(\beta)$
for each word position $i, j$ do
    (a) Choose a topic $z_{i,j} \sim \mathrm{Multinomial}(\theta_i)$.
    (b) Choose a word $w_{i,j} \sim \mathrm{Multinomial}(\varphi_{z_{i,j}})$
end
```

Algorithm 1 : Generative Model

The generative model contains the main ideas of LDA. Data is treated as observations coming from a process which samples words from a (hidden) distribution over topics. Every word is generated independently from any other word. The generative process does not take word order into account. Hence, the generative model produces a bag-of-words, where word order does not play a role.
If 3 topics are chosen, e.g. *Linguistics, Philosophy* and *Computer Science*, and if equal probability is given to all of them, then the generative process might produce bag-of-words resembling a text from *Digital Humanities.* By putting more weight on *Linguistics* and *Computer Science* the result would resemble documents from the domain of *Computer Linguistics.* The generative process of LDA is depicted as a Bayesian probabilistic model in Figure 4.

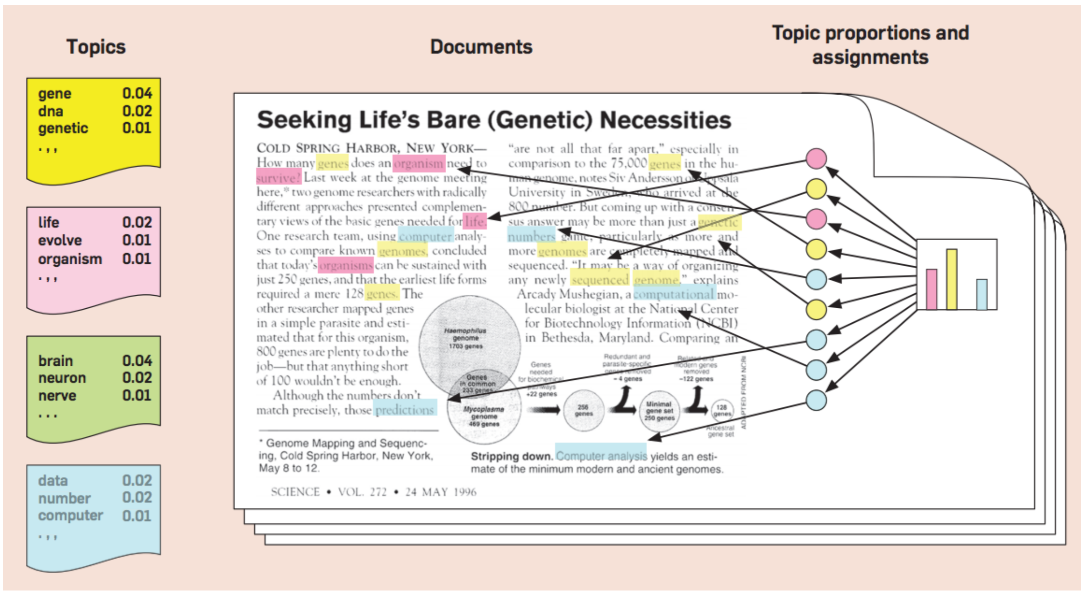


Figure 3 The generative process. Every word originates from a topic coming from a distribution over topics. The topics and their word distributions are depicted on the far left. E.g. the *yellow* topic is made up of the word *gene* by 4%. Taken from [1]

.

In Figure 3 the generative model is illustrated. The topic proportions are plotted as a bar chart, the colored coins are the topic assignments for each word.

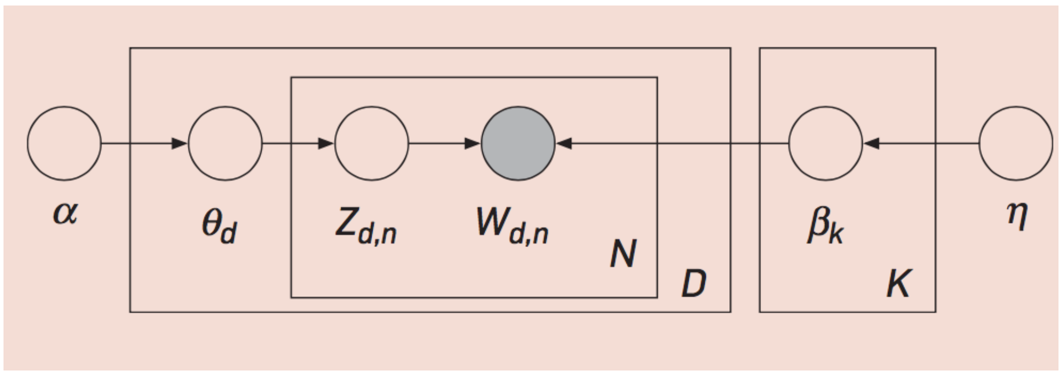


Figure 4 Plate notation for Latent Dirichlet Allocation corresponding to the generative process. Every vertex depicts a random variable. The shaded node $w_{d,n}$ stands for the observed words. Everything else is not observed and must be inferred. Taken from [1]

.

1.5 Experiments

To fully grasp what LDA will produce as output, this section will provide an example with *real world* data.
Gensim[1] is a tool for unsupervised semantic modeling. The idea is to define three topics by using Wikipedia articles. Then, all those Wikipedia articles are put into one corpus to see whether LDA can reproduce those three original topics.
With Gensim the model can be trained very easily[2]:

Listing 1 Query for a document

```
K = 3
lda = ldamodel.LdaModel(CORPUS, id2word=dictionary, num_topics=K,
update_every=1, chunksize=1, passes=350)
```

`CORPUS` contains the following preprocessed Wikipedia articles. The preprocessing steps comprise removing stop words and transforming the documents to a bag-of-words.

• *Computer science*	Algorithm, Computation, Computer Programming, Programming Language, Computational complexity theory, Computability theory, Artificial Intelligence
• *Philosophy*	Epistemology, Metaphysics, Continental philosophy, Ancient Greek, Ethics, Aesthetics, Art, Phenomenology (philosophy), Pythagoras, Plato
• *Linguistics*	Language, Semantics, Syntax, Phonology, Grammar, Phonetics, Pragmatics, Corpus linguistics, Linguistic prescription, Linguistic description

[1] https://radimrehurek.com/gensim/index.html
[2] the complete ipython notebook can be found here: https://github.com/flobeck/algo-seminar/blob/master/ipython-notebook/lda_mixture.ipynb

The output of a call of LDA with `K=3` is summarized in table 2.

Table 2 Each of the three topics is a distribution over the words. E.g. topic 1 resembles *computer science*. The table only shows the most relevant words for each topic.

TOPIC	REL. FREQUENCIES	WORD
1	0.010	programming
	0.009	turing
	0.008	problem
	0.007	machine
	0.006	problems
	0.006	algorithms
	0.006	algorithm
	0.006	set
	0.006	complexity
	0.005	time
2	0.008	plato
	0.006	plato's
	0.006	knowledge
	0.005	pythagoras
	0.005	philosophy
	0.004	phenomenology
	0.004	greek
	0.004	according
	0.004	socrates
	0.003	theory
3	0.019	language
	0.009	languages
	0.007	art
	0.005	meaning
	0.005	linguistic
	0.005	human
	0.004	ethics
	0.004	speech
	0.004	study
	0.004	grammar

Given a set corpus of Wikipedia articles, LDA finds three topics which resemble the topics *computer science*, *philosophy* and *linguistics*.
Furthermore, it is possible to query where on the topic simplex an unseen document is located:

Listing 2 Query for a document

```
w = wikipedia.page("Noam Chomsky").content
bow_vector = dictionary.doc2bow(tokenize(w))
print lda[bow_vector]

>> [(1, 0.032599745516170064),
```

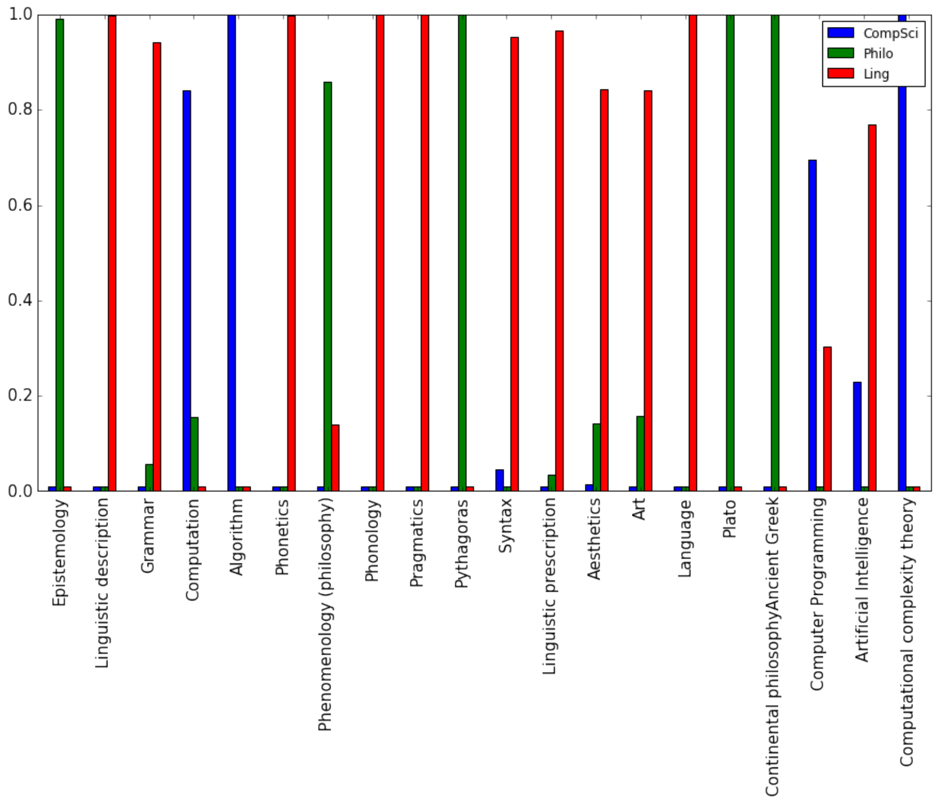


Figure 5 Topic proportions for some of the Wikipedia articles that were used to train the model. The Wikipedia article on 'Pythagoras', for instance, is classified as a document in the category 'philosophy' with very high confidence. Note that albeit the confidence is very high it is not 100%. Due to the model every document exhibits all the K topics, but to a very different degree.

```
(2, 0.30504589551791855),
(3, 0.66235435896591133)]
```

In this case (Listing 2), the Wikipedia article on the linguist *Noam Chomsky* is made up of 66% Topic 3 (*Linguistics*), 30% Topic 2 (*Computer Science*) and 3% Topic 1 (*Philosophy*).

1.6 (Bayesian) Inference

Inference means computing the *posterior* and corresponds to '*reversing*' the generative model. In other words, inference is about fitting a generative model (or rather the hidden parameters), such that it explains the observed data.

Many methods solving the problem of maximum a posteriori estimation have been proposed. The original paper on LDA [2] suggested a Variational Bayesian method which can be understood as an extension of the expectation-maximization algorithm [3]. Gibbs Sampling as a way to compute the posterior was proposed by Steyvers and Griffiths [7]

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} | w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})} \tag{9}$$

The posterior is intractable to compute. More precisely, the denominator of (Equation 9), the marginal probability, is not feasible to compute. To compute $p(w_{1:D})$ one would have to sum up the joint distribution over all possible instances of the hidden topic structure.

1.6.1 Gibbs Sampling

Gibbs Sampling, named after the physicist Josiah Willard Gibbs, is a Markov Chain Monte Carlo Algorithm (MCMC) and can be used to approximate the marginal and posterior distribution by constructing a Markov chain which converges to the target distribution. A Markov chain represents a random process, where going from one state to another is determined by a transition matrix T. An entry $p_{i,j}$ corresponds to the probability of going from state i to state j. The goal of Gibbs Sampling is to integrate out the per-document topic proportions θ. In order to assign a word to a topic, the Gibbs sampler computes the probability of a topic $z_{d,n}$ if it is assigned to a word $w_{d,n}$, where all other word-to-topic assignments are given or fixed. $z_{\_d,n}$ is defined to be the notation for all topic assignments except for for $z_{d,n}$
Formally:

$$p(z_{d,n} = k | z_{\_d,n}, w, \alpha, \beta) = \frac{p(z_{d,n} = k, z_{\_d,n} | w, \alpha, \beta)}{p(z_{\_d,n} | w, \alpha, \beta)} \tag{10}$$

It was shown [4] that Equation 10 can be calculated by:

$$p(z_{d,n} = k | z_{\_d,n}, w, \alpha, \beta) \propto \frac{C^{WT}_{n,k} + \beta}{\sum_{n=1}^{W} C^{WT}_{n,k} + W\beta} \cdot \frac{C^{DT}_{d,k} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d,t} + T\alpha} \tag{11}$$

$C^{WT}_{n,k}$ and $C^{DT}_{d,k}$ are matrices. $C^{WT}_{n,k}$ is the number of times topic k was assigned to word n. On the other hand $C^{DT}_{d,k}$ is the number of times topic k was assigned to words in document

d. The first step of the Gibbs Sampling algorithm is to randomly assign every word of all documents to a topic. By this the count matrices $C^{WT}_{n,k}$ and $C^{DT}_{d,k}$ are filled with values. Gibbs Sampling will now sample topic assignments according to Equation 11. The first term of the product in Equation 11 also corresponds to an estimate of φ_i, and the second term to an estimate of θ_j

$$\hat{\varphi}_i = \frac{C^{WT}_{i,j} + \beta}{\sum_{k=1}^{W} C^{WT}_{k,j} + W\beta} \tag{12}$$

$$\hat{\theta}_j = \frac{C^{DT}_{d,j} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d,t} + T\alpha} \tag{13}$$

▶ **Example 1.** The following step-by-step example illustrates how Gibbs Sampling can be used to assign a topic to a word. Consider the sentence (which represents one document out of a corpus):

Text mining algorithms can be used to find structure in text corpora like Plato's dialogues

A priori we choose the number of topics $K = 3$
The first step is to remove stop words and randomly assign each word to a topic (Table 3). This is done to all documents.

Table 3 random initialization of topic assignments to words.

1	3	2	1	2	1	2
text	mining	algorithms	structure	corpora	Plato	dialogues

The counts for all documents is shown in Table 4.

Table 4 word counts for every topic

	1	**2**	**3**
text	65	54	59
mining	21	4	12
algorithms	100	74	122
structure	20	12	14
corpora	5	2	12
Plato	35	33	42
dialogues	24	27	31

The goal is now to resample the word *algorithm*.

Table 5 We first choose *algorithm* to be resampled.

1	3	???	1	2	1	2
text	mining	algorithms	structure	corpora	Plato	dialogues

The assignment of the word *algorithm* is determined by sampling according to the topic distribution in the document and the word distribution over the corpus. In Figure 6 the sampling is depicted. The blue bars correspond to the topic distribution, the yellow bars to the word distribution over topics. Accordingly, it is most likely that *algorithm* is assigned to topic 1, since the green area for topic 1 is the largest. After assigning the word, the word counts for each topic get updated and the first iteration of the Gibbs sampling algorithm is completed. This process is repeated several times, until the assignments do not change anymore.

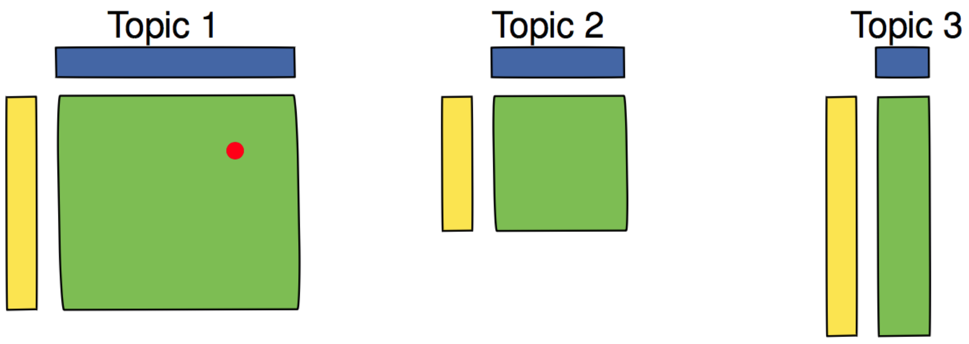


Figure 6 Sampling according to the green area.

1.7 Conclusion

Topic models enable scholars to find the hidden topic structure in large text corpora. The hidden structure must be uncovered in order to gain information about the low-dimensional structure behind a text corpus. A topic modeling algorithm takes a corpus as an input and will output the topics running through that corpus. The generative process is the essential idea behind Latent Dirichlet Allocation. Words come from a distribution over topics, which are again distribution over words. Reversing the generative model by applying Gibbs sampling yields a solution of the inference problem, which can not be solved analytically.

References

1 David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

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

3 Charles W Fox and Stephen J Roberts. A tutorial on variational bayesian inference. *Artificial intelligence review*, 38(2):85–95, 2012.

4 Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

5 Michael J Paul and Mark Dredze. You are what you tweet: Analyzing twitter for public health. *ICWSM*, 20:265–272, 2011.

6 Pietro Pinoli, Davide Chicco, and Marco Masseroli. Latent dirichlet allocation based on gibbs sampling for gene function prediction. In *Computational Intelligence in Bioinformatics and Computational Biology, 2014 IEEE Conference on*, pages 1–8. IEEE, 2014.

7 Mark Steyvers and Tom Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.