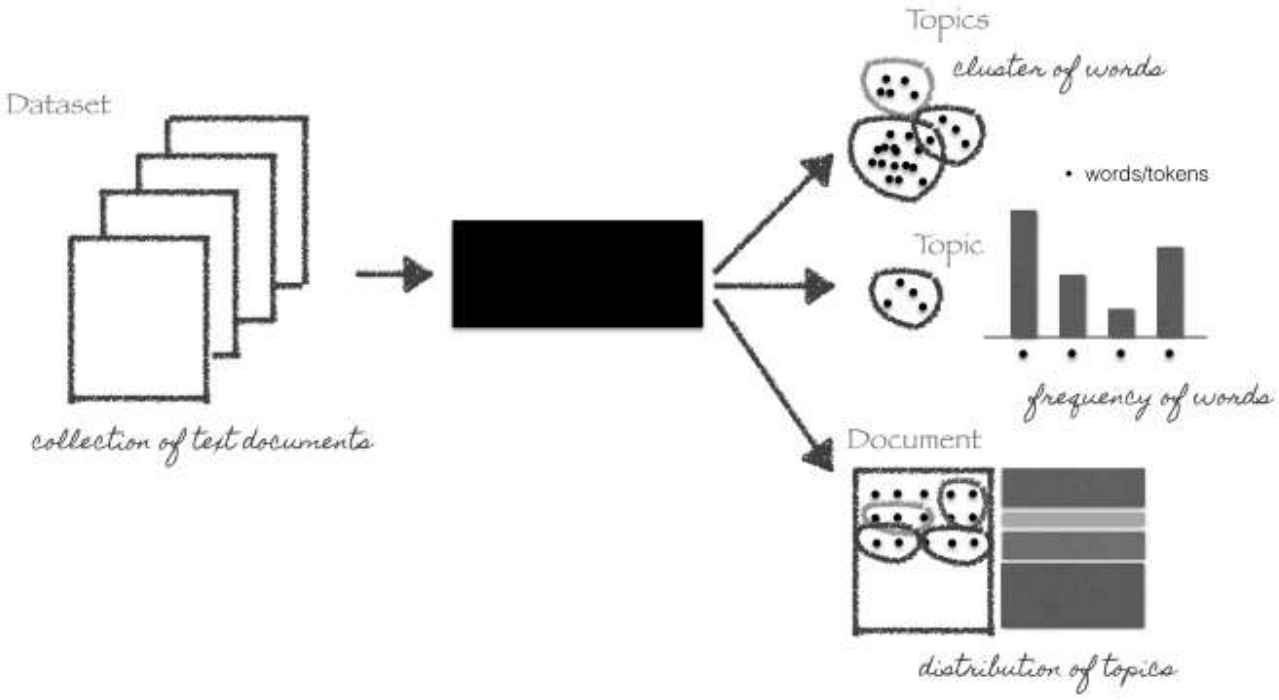
Topic modeling using Latent Dirichlet Allocation(LDA) and Gibbs Sampling explained!

Ankur Tomar Follow
Nov 25, 2018 · 9 min read



Credits: Christine Doig

Hi everyone. It's been a long time since I wrote my last blog on the different type of recommender systems. This time, I will be writing about Topic Modelling.

There are numerous sources out there where you can learn what Topic Modelling is, but I think, most of them are somewhat technical and requires a good knowledge of math and other techniques like Markov Chains. In this article, I will try to explain the idea

behind Topic Modelling in a very intuitive way with bare minimum math and technicalities. I will write this article in two parts. In the first part, I will try to explain what topic modeling is and in the second part, I will provide a python tutorial of how to do topic modeling on the real-world dataset.

Before I start, I want to acknowledge that this article is heavily inspired by the lecture of my exploratory data analytics Prof. Edward McFowland, Coursera's course on Natural Language Processing by Higher School of Economics and Jordan Boyd-Graber explanation of Gibbs sampling.

INTRODUCTION

What is topic modeling?

Topic modeling is a branch of unsupervised natural language processing which is used to represent a text document with the help of several topics, that can best explain the underlying information in a particular document. This can be thought in terms of clustering, but with a difference. Now, instead of numerical features, we have a

collection of words that we want to group together in such a way that each group represents a topic in a document.

Why do we need topic modeling?

Okay, so now the question arises why do we need topic modeling? If we look around, we can see a huge amount of textual data lying around us in an unstructured format in the form of news articles, research papers, social media posts etc. and we need a way to understand, organize and label this data to make informed decisions. Topic modeling is used in various applications like finding questions on stack overflow that are similar to each other, news flow aggregation and analysis, recommender systems etc. All of these focus on finding the hidden thematic structure in the text, as it is believed that every text that we write be it a tweet, post or a research paper is composed of themes like sports, physics, aerospace etc.

How to do topic modeling?

Currently, there are many ways to do topic modeling, but in this post, we will be discussing a probabilistic modeling approach called Latent Dirichlet Allocation (LDA) developed by Prof. David M. Blei in 2003. This is an extension of Probabilistic Latent Semantic Analysis (PLSA) developed in 1999 by Thomas Hoffman with a very minute difference in terms of how they treat per-document distribution. So let's jump straight into how LDA works.

Latent Dirichlet Allocation

Let's start by understanding the meaning of each word in the title, as I think it contains everything that we need to know to understand how LDA works.

Latent: This refers to everything that we don't know a priori and are hidden in the data. Here, the themes or topics that document consists of are unknown, but they are believed to be present as the text is generated based on those topics.

Dirichlet: It is a 'distribution of distributions'. Yes, you read it right. But what does this mean? Let's think about this with the help of an example. Let's suppose there is a machine that produces dice and we can control whether the machine will always produce a dice with equal weight to all sides, or will there be any bias for some sides. So,

the machine producing dice is a distribution as it is producing dice of different types. Also, we know that the dice itself is a distribution as we get multiple values when we roll a dice. This is what it means to be a distribution of distributions and this is what Dirichlet is. Here, in the context of topic modeling, the Dirichlet is the distribution of topics in documents and distribution of words in the topic. It might not be very clear at this point of time, but it's fine as we will look at it in more detail in a while.

Allocation: This means that once we have Dirichlet, we will allocate topics to the documents and words of the document to topics.

That's it. This is what LDA is in a nutshell. Now let's understand how this works in topic modeling.

Just to recap, what LDA says is that each word in each document comes from a topic and the topic is selected from a per-document distribution over topics. So we have two matrices:

1. $\theta td = P(t|d)$ which is the probability distribution of topics in documents

2. $\Phi wt = P(w|t)$ which is the probability distribution of words in topics

And, we can say that the probability of a word given document i.e. $P(w|d)$ is equal to:

$$\sum_{t \epsilon T} p(w|t,d)\, p(t|d)$$

where T is the total number of topics. Also, let's assume that there is W number of words in our vocabulary for all the documents.

If we assume conditional independence, we can say that

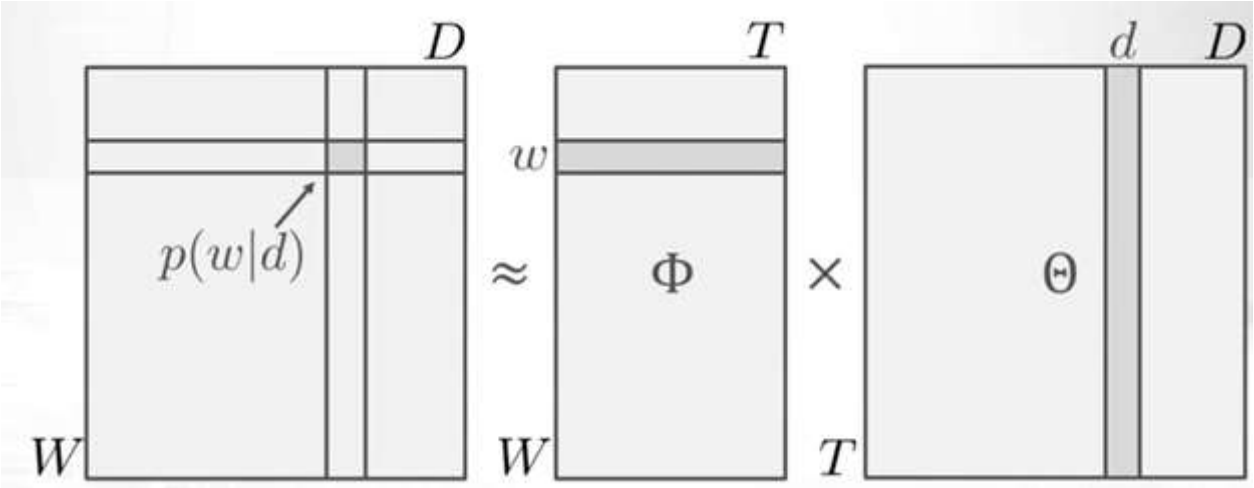
$P(w|t,d) = P(w|t)$

And hence $P(w|d)$ is equal to:

$$\sum_{t=1}^{T} p(w|t)\, p(t|d)$$

that is the dot product of Θtd and Φwt for each topic t.

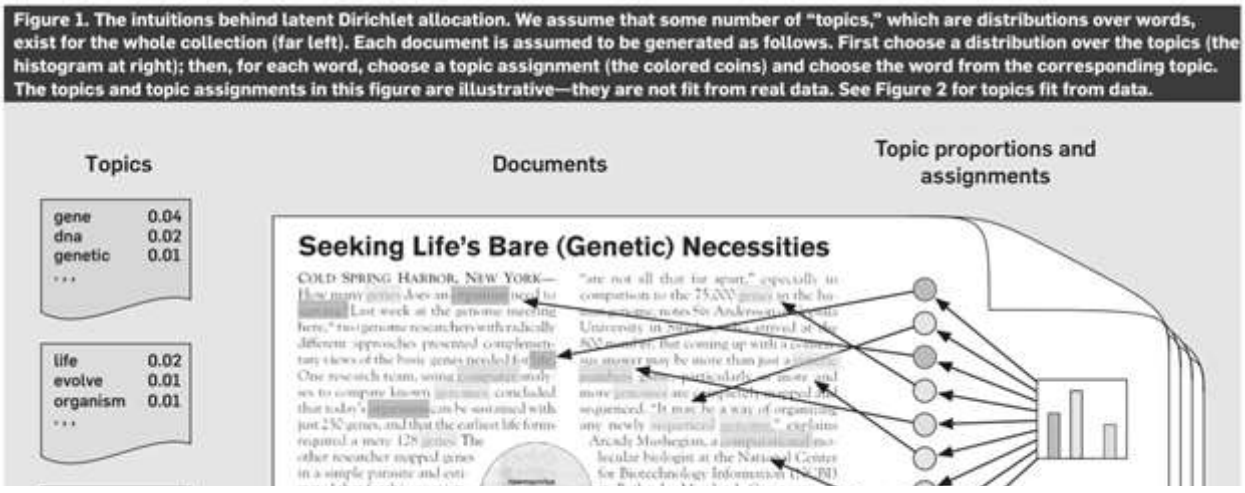
This can be represented in the form of a matrix like this:

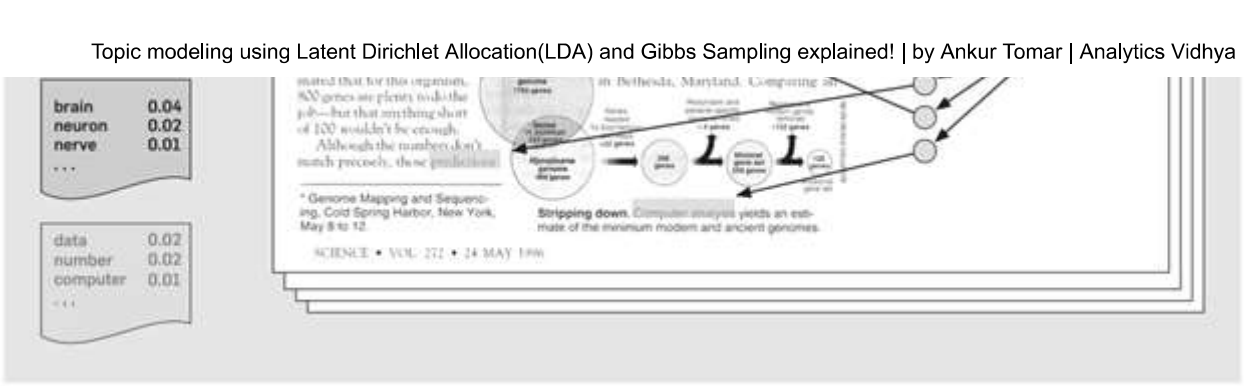


Credit: Higher School of Economics

So, looking at this we can think of LDA similar to that of matrix factorization or SVD, where we decompose the probability distribution matrix of word in document in two matrices consisting of distribution of topic in a document and distribution of words in a topic.

Therefore, what we will get is, for example, this:





Credit: David M. Blei

And to tie back to our example of dice, we can say that each word in the distribution of words in a topic is similar to a side of the dice, and we have Dirichlet parameter to control if all the words have same probability in a topic or will that topic have an extreme bias towards some words. Same intuition is for distribution of topics in a document.

Good. Now comes the important part. How do we learn the weights of these two matrices?

To start with, let's randomly assign weights to both the matrices and assume that our data is generated as per the following steps:

1. Randomly choose a topic from the distribution of topics in a document based on their assigned weights. In the previous example, let's say we chose pink topic

2. Next, based on the distribution of words for the chosen topic, select a word at random and put it in the document

3. Repeat this step for the entire document

In this process, if our guess of the weights is wrong, then the actual data that we observe will be very unlikely under our assumed weights and data generating process. For example, let's say we have document D1 which consists of the following text:

"*Qualcomm® Adreno™ 630 Visual Processing Subsystem featuring room-scale 6DoF with SLAM, Adreno Foveation*"

and let's say we assign high weights to topic T1 which has high weights for words like Spoon, Plates, Onions etc. then we will see that given our assumption of how data is

generated, it is very unlikely that T1 belongs to D1 or these words belongs to T1. Therefore, what we are doing is we are trying to maximize the likelihood of our data given these two matrices.

To identify the correct weights, we will use an algorithm called Gibbs sampling. Let's now understand what Gibbs sampling is and how does it work in LDA.

Gibbs Sampling

Gibbs sampling is an algorithm for successively sampling conditional distributions of variables, whose distribution over states converges to the true distribution in the long run. This is somewhat an abstract concept and needs a good understanding of Monte Carlo Markov Chains and Bayes theorem. These concepts and the math behind them is fairly complex and is out of the scope of this blog. Here, I will try to give an intuition on how Gibbs Sampling work to identify topics in the documents.

As I mentioned earlier, to start with, we will assume that we know Θ and Φ matrices. Now what we will do is, we will slowly change these matrices and get to an answer that maximizes the likelihood of the data that we have. We will do this on word by word basis by changing the topic assignment of one word. We will assume that we don't know the topic assignment of the given word but we know the assignment of all other words in the text and we will try to infer what topic will be assigned to this word.

If we look at this in mathematical manner, what we are doing is we are trying to find conditional probability distribution of a single word's topic assignment conditioned on the rest of the topic assignments. Ignoring all the mathematical calculations, what we will get is a conditional probability equation that looks like this for a single word *w* in document *d* that belongs to topic *k*:

$$p(z_{d,n} = k|\vec{z}_{-d,n}, \vec{w}, \alpha, \lambda) = \frac{n_{d,k} + \alpha_k}{\sum_i^K n_{d,i} + \alpha_i} \frac{v_{k,w_{d,n}} + \lambda_{w_{d,n}}}{\sum_i v_{k,i} + \lambda_i}$$

Credit: Jordan Boyd-Graber

where:

n(d,k): Number of times document d use topic k

v(k,w): Number of times topic k uses the given word

αk: Dirichlet parameter for document to topic distribution

λw: Dirichlet parameter for topic to word distribution

There are two parts two this equation. First part tells us how much each topic is present in a document and the second part tells how much each topic likes a word. Note that for each word, we will get a vector of probabilities that will explain how likely this word belongs to each of the topics. In the above equation, it can be seen that the Dirichlet parameters also acts as smoothing parameters when *n(d,k)* or *v(k,w)* is zero which means that there will still be some chance that the word will choose a topic going forward.

Let's go through an example now:

To start with, suppose we have a document with some random word topic assignment, for example, as shown below:

India	enters	world	cup	final
1	3	1	2	4

We also have our count matrix v(k,w) as shown below:

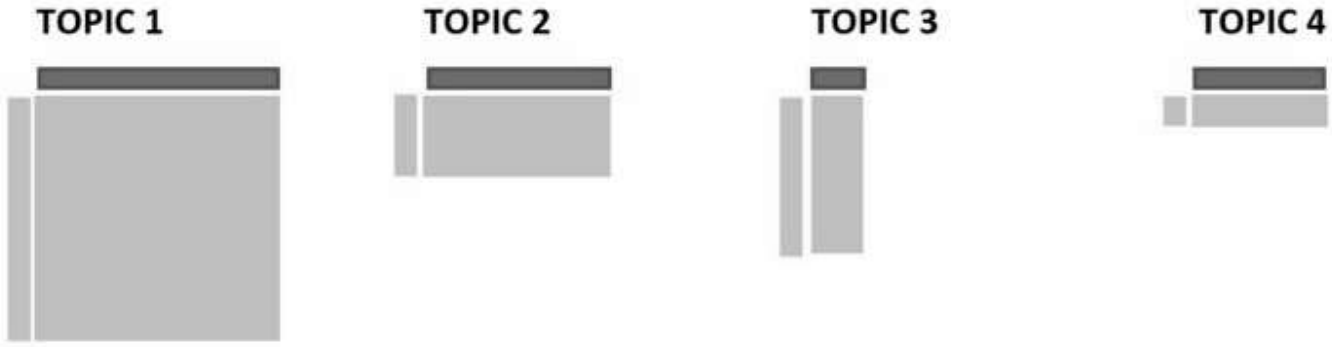
	1	2	3	4
India	70	5	0	8
enters	2	3	15	6
world	28	4	12	1
cup	6	43	6	0
final	7	0	9	31

Now let's change the assignment of word **world** in the document.

- First, we will reduce the count of world in topic 1 from 28 to 27 as we don't know to what topic world belongs.
- Second let's represent the matrix *n(d,k)* in the following way to show how much a document use each topic

· Third, let's represent *v(k,w)* in the following way to show how many times each topic is assigned to this word

- Fourth, we will multiply these two matrices to get our conditional probabilities



- Finally, we will randomly pick any of the topic and will assign that topic to **world** and we will repeat these steps for all other words as well. Intuitively, topic with highest conditional probability should be selected but as we can see other topics also have some chance to get selected

That's it. This what Gibbs sampling algorithm is doing under the hood. Although we skipped some details like hyperparameter tuning, but from an intuition perspective, this is how Gibbs sampling works for topic modeling.

So, this is it for the theory of Latent Dirichlet Allocation. I hope this was helpful in understanding what topic modelling is and how we use LDA with Gibbs for topic modelling. In the next article, I will post the implementation of LDA using Python.

Thanks!