

# Dissertation

Ryan Pollard

03/08/2021

## Contents

<b>1. Text Analysis and NLP: A Background</b>	<b>2</b>
1.1 How computers use text data . . . . .	2
1.2 Applications . . . . .	3
1.3 Supervised and Unsupervised Machine Learning . . . . .	3
<b>2. Topic Models</b>	<b>4</b>
2.1 Introduction . . . . .	4
2.2 Methods for topic modeling . . . . .	4
2.3 Latent Semantic Indexing . . . . .	4
2.4 Non-negative matrix factorization . . . . .	4
<b>3. Latent Dirichlet Allocation</b>	<b>5</b>
3.1 LDA Introduction . . . . .	5
3.2 LDA In-Depth . . . . .	5
3.2 The Generative Process of LDA . . . . .	6
3.2.1 Notation . . . . .	7
3.3 The Dirichlet distribution . . . . .	7
3.3.1 Dirichlet in detail . . . . .	9
Plate notation . . . . .	9
3.4 Gibbs sampling . . . . .	11
3.4.1 Gibbs Sampling in detail . . . . .	11
Obtaining the topic and word distributions. . . . .	12
3.5 Benefits of LDA . . . . .	12
<b>References</b>	<b>13</b>



# 1. Text Analysis and NLP: A Background

Data is increasingly becoming more important and more available to businesses and individuals alike. Businesses have so much data and with that comes the difficulty to manage that data. For example, a business might sell their goods on Amazon and receive thousands of customer reviews a day. We traditionally think of data as being spreadsheets of numbers however this textual-review data represents an indication, a piece of data, on the quality of that product. In the same way as reviews, social media represents a vessel of data that represents the thoughts and feelings of its users. From casually browsing twitter for example, we can read a few tweets regarding the latest news event and after reading 5 to 10 tweets we could probably conclude how the users of twitter are feeling about this event.

That browsing and reading of tweets is essentially a data gathering exercise to discern the opinion of the people. Since language is the way humans interface and naturally share data, it's completely natural to humans to take these words on a screen from multiple tweets and form a single opinion about how people must be feeling about that event. For example, looking at tweets that speak about tax increases, we can read 10 tweets, 7 of which may criticize this change in policy, 2 of which may praise it, and 1 that is in between those two feelings. From this a reader of twitter can tell that the attitude to increasing taxes is largely negative.

With the advent of smartphones across developed and developing countries, the ease of access of internet, the volume of voluntarily inputting text data has exploded.

In terms of analysis this data, the problem with text data is it is unstructured, unlike numbers that represent financials or demographics with set numerical rules; and therefore it is inherently difficult to work with. Text often is hard to interpret to a machine, with its grammar, syntax rules and various patterns and as a result it isn't straight forward to carry out statistical techniques on text data.

There are however rigorous statistical techniques that have been developed that lets a machine interpret with text data. Under the umbrella known as "NLP" - Natural Language Processing, these techniques aim to understand text-based data.

Natural Language Processing, where a Natural Language is any language that evolved with humans to communicate, uses the linguistic rules that every language has to extract information from pieces of text. It enables computers to process and understand human natural language.

## 1.1 How computers use text data

First a definition, a "document" is a single separate piece of text. In the context of tweets, a document would be one tweet. If we were analysing news articles, a document would be one article. Documents comprise a "corpus" which is just all the documents; our entire dataset.

Machine learning techniques need numeric data. A logistic regression is a simple machine learning tool that can express the segmentation of a dependent variable with two groups as a function of its independent variables. With text data we have exactly the same framework.

Perhaps we have 10 documents (let's say a document is one tweet, in this example), 5 of which give misinformation about covid and 5 of which give true information about covid. We want to create a logistic regression such that we know the relationship between the text in each tweet and its label of misinformation/information.

The logistic regression cannot do this, because all the machine learning algorithms can only interpret numbers. The logical step then is to convert these documents (tweets, in our example) into numbers. This is done via vectorisation our document, and all the documents then is represented as a matrix where each row is a vector that represents the text in one of the documents. Using this matrix, we can then undertake machine learning techniques on our text data.

First, each document (piece of text) is split so each word in the document is a data point. This is called tokenisation which is just a list of words that appear in each document. Then for each document, this list of words is vectorised using one of the following techniques:

1. Bag of words - a matrix where each word is a column, each row is a document (one piece of text data) and the number in each cell is the number of times the column's word appears in the row's text. See the image below for an example.

	about	bird	heard	is	the	word	you
About the bird, the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

Figure 1: Figure caption

2. TF-IDF - stands for "Term Frequency — Inverse Document Frequency." This generates a matrix that assigns a weight to each word which signifies the importance of the word in the document and corpus. This is more sophisticated than the above as the frequency the word appears used across all the texts is used to assign an importance.
3. word2vec - uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. This is even more sophisticated than the above as context of the word is taken into account.

Using these matrices, machine learning techniques can be used in order to classify documents and other operations.

## 1.2 Applications

- **Machine Translation** - translating text from one language to another. Becoming more accurate with the aid of Deep Learning.
- **Speech Recognition**
- **Question Answering Systems** - For example chatbots and personal assistants such as Siri
- **Contextual Recognition** - Getting meaning of words from the context of the sentence, not just using the definition of the word.
- **Text Summarisation** - Taking a large piece of text and condensing it but retaining the original meaning .
- **Text Categorisation** - Classifying texts. For example, perhaps given a piece of text, a tweet in this example, analysing the words we could classify it as a tweet that contains misinformation about covid to a certain level of accuracy using machine learning techniques.
- **Text Analytics** - Deriving insights from text data. Methods include clustering, summarisation, sentiment analysis. An example application of this is Spam detection. Statistical techniques are used to classify certain emails as spam.

## 1.3 Supervised and Unsupervised Machine Learning

Supervised machine learning is taking already labeled data and using it to express the relationship between the labels and the independent variables in the data. In our running example it would mean taking twitter data in regards to covid and manually labeling them "misinformation" is it contains misinformation and "information" is it contains real information. We could then explore the relationship between the text and


the labels and use this relationship to predict if a new tweet that hasn't been labeled as having misinformation to a certain percent of accuracy.

However, since there is exponentially more and more text information available it is impossible to label the data in this way. Therefore it is common to apply unsupervised (unlabeled) statistical techniques to text data. These refer to the family of machine learning algorithms that try to discover latent hidden structures and patterns in data from their various attributes and features. Several unsupervised learning algorithms are also used to reduce the feature space, which is often of a higher dimension to one with a lower dimension. This dimensionality reduction can be seen as taking a large number of **document** and assigning them into a relatively small amount of groups, where each group has its own topic **where the topic** describes broadly the type of tweets the group contains.

## 2. Topic Models

### 2.1 Introduction

Imagine we have 20,000 tweets that express opinions about **Covid**. Using unsupervised machine learning and the vectorisation of text explained above so we can mathematically express and analyse the text in each tweet, we could group these tweets into 3 groups, where the tweets in one group 1 would contain text that speaks about **covid** cures, group 2 would have tweets about **covid** vaccines and group 3 about covid prevention. To reach this goal, we use a range of techniques called Topic Models.

Topic models extract the distinguishing concepts, or topics, from the set of words in the corpus (that is, all the documents). **They allow us**, without human intervention, to group up documents quickly into topics. These topics can include opinions or facts. The models then use statistical techniques to explore the hidden and latent structures in the corpus in order to group up documents. 


### 2.2 Methods for topic modeling

There are various algorithms to carry out Topic Modelling.

**We cover** the following three methods: \* Latent Semantic Indexing \* Latent Dirichlet Allocation \* Non-negative matrix factorization

#### 2.3 Latent Semantic Indexing

Latent Semantic Analysis simply finds groups of documents with the same words. The **LSA** approach to topic modeling (also known as Latent Semantic Indexing) identifies themes within a corpus by creating a sparse term-document matrix, where each row is a token and each column is a document. Each value in the matrix corresponds to the frequency with which the given term appears in that document. Singular Value Decomposition (SVD) can then be applied to the matrix to factorise into matrices that represent the term-topics, the topic importances, and the topic-documents.

Using the derived diagonal topic importance matrix, **we can** identify the topics that are the most significant in **our corpus**, and remove rows that correspond to less important topic terms. Of the remaining rows (terms) and columns (documents), **we can** assign topics based on their highest corresponding topic importance weights. 

#### 2.4 Non-negative matrix factorization

One way to find the hidden topics of a corpus is the factorization of the “document-term matrix” - this is the a matrix with the rows being the documents and the columns being the words in the whole corpus - and if a document contains a particular word in the corpus it gets a value of 1 in that cell. This matrix has only positive-value elements and so we can use methods from linear algebra that allow us to represent the matrix as the product of two other nonnegative matrices. The original matrix is called  $V$ , and the factors are  $W$  and  $H$ :

$$V \approx W \cdot H$$



In the context of text analytics, both  $W$  and  $H$  have an interpretation. The matrix  $W$  has the same number of rows as the document-term matrix and therefore maps documents to topics (document-topic matrix).  $H$  has the same number of columns as features, so it shows how the topics are constituted of features (topic-feature matrix). The number of topics (the columns of  $W$  and the rows of  $H$ ) can be chosen arbitrarily. The smaller this number, the less exact the factorization.

### 3. Latent Dirichlet Allocation



#### 3.1 LDA Introduction

LDA considers each document as consisting of different topics, so each document is a mix of different topics. Also, the topics are made up of words. To ensure that the number of topics per document is low and to have only a few important words constituting the topics, LDA uses a Dirichlet distribution, a Dirichlet prior. This is applied both for assigning topics to documents and for assigning words to the the topics.

After the initial assignments, a generative process begins. It uses the Dirichlet distributions for topics and words and tries to re-create the words from the original documents with stochastic sampling. This process has to be iterated many times and is therefore computationally intensive. On the other hand, the results can be used to generate documents for any identified topic. For a given corpus of documents, each document can be represented as a statistical distribution of a fixed set of topics. LDA assumes that each document is generated by a statistical generative process. That is, each document is a mix of topics, and each topic is a mix of words.

#### 3.2 LDA In-Depth

Topic models are based on the idea that each document, that is each piece of text, are a mixture of topics. For example, perhaps there is a news article about electric vehicles and how it impact the economy. In this example it can be said this document (i.e the article) is a combination different topics, not just one - it could well be 50% transport, 30% technology and 20% economy. We label this distribution We call this  $\theta$ .

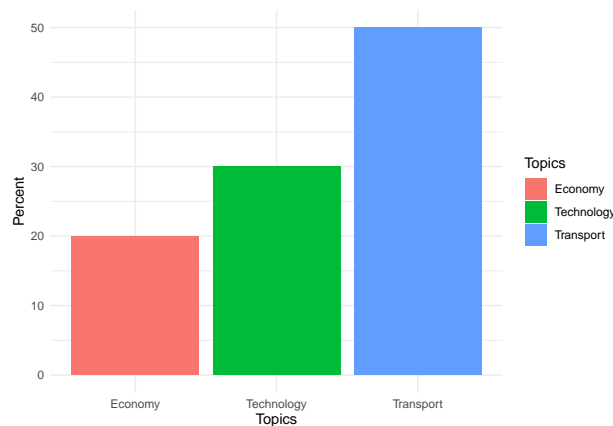


Figure 2: Example of a multinomial distribution of the Topics in LDA.

We then say that the documents have a probability distribution over topics, as described in the example. A document chosen at random may have any percent of  $n$  different topics.

Equally, it can be said a topic is a distribution over words. For example, the topic transport can be defined by a list of words that appear in that topic with varying frequencies. For example, perhaps the topic transport,

for simplicity, involved three words - car, bus, roads. Perhaps car appears 70% of the time for that topic, bus 10% and roads 20%. Figure 3 shows a toy example of a word distribution for the topic Transport in from Figure 2. The other 2 topics in Figure 2 would have their own word distributions.

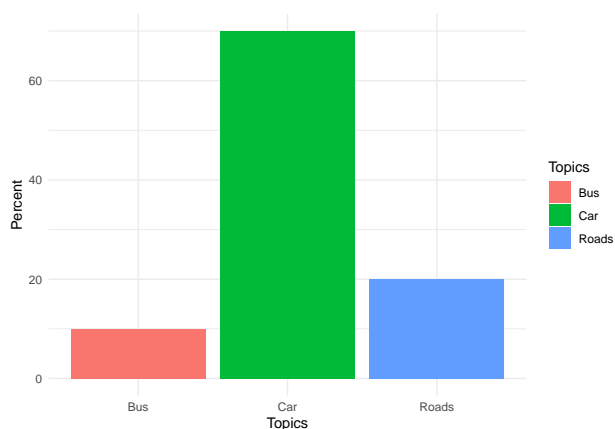


Figure 3: Example of a multinomial distribution of the words in a Topic in LDA.

This is analogous to the reality where, if you were to read an article you might decide what topic it's about by the words used in the article - with some words associated with that topic appearing more often (for example car would appear more often than petrol station across all transport documents.), the difference is in reality the amount of words in each topic is extremely large.

A topic model is a generative process. The model describes a statistical procedure through which the documents we observe are produced. A generative process for documents describes how words in documents might be generated based on variables we don't see - latent variables. It aims to find the best set of latent variables that explains the documents in the corpus. These latent variables dictate the probability distributions above.

### 3.2 The Generative Process of LDA

One document in our corpus can be assumed to be created by first choosing a distribution over topics - just like how an author might decide he is going to write about 60% "transport" and 40% "technology," these two topics are then the distribution of topics per document. After, for each word in the document, the author chooses a word following the distribution of words from a chosen topic which is sampled from the distribution of topics per document. For example perhaps the topic "transport" is chosen for the first word, and within transport there is a distribution of words that looks like 60% "car" and 40% "bus." A word is chosen from this distribution and since car is most likely, for this first word "car" is chosen.

The role of Probabilistic Topic Models is inferring the topics through which the documents we have are generated.

In practice when using LDA each document is a distribution over many topics, but typically each document is dominated by one or two topics, just as in real life an article typically only has one or two topics. Each of those topics have also a distribution of words, and typically many words. For example the topic may have 1000s of words related to that topic.

The advantage of this method is that each topic is interpretable by seeing the distribution of words in that topic, and therefore the topic can be given a name just by seeing this distribution. For example, if the words "fight" and "glove" were grouped together in a distribution with 80% and 20% probability respectively, we can easily say that this topic is about boxing.

### 3.2.1 Notation

The distribution over words within the document is specified by the model and is given by:

$$P(w_i) = \sum_{j=1}^T P(w_i|z_j = j)P(z_i = j) \quad (1)$$

Where:

- $P(z_i = j)$ , the probability that the  $j$ th topic was chosen for the  $i$ th word
- $P(w_i|z_j = j)$ , the probability of the word  $w_i$  under topic  $j$
- $T$ , the number of topics.



And let:

- $\phi^{(j)} = P(z = j)$ , a multinomial distribution over words for topic  $j$
- $\theta^{(d)} = P(z)$ , a multinomial distribution over topics for document  $d$
- $D$ , the number of documents in the corpus
- $N_d$ , number of words in each document  $d$
- $N$ , total number of words in the corpus
- *Word Token*, the assignment of a word to a place in the document where there is a word present.



The parameter  $\theta$  indicates what topics are important for a particular document and  $\phi$  indicates what words are important in each topic.

### 3.3 The Dirichlet distribution

Multinomial distributions can be drawn from a Dirichlet distribution, which we need to estimate  $\phi$  and  $\theta$ , our latent variables which ultimately LDA aims to approximate. In Figure 5 below we can see changing the parameter  $\alpha$ , which represents the variance of the Dirichlet distribution (that also differs the the multinomial distributions variance). In the equation in 4 below,  $m_k$  represents the mean and is analogous to centre of the Dirichlet distribution. When  $\alpha * m = 1$  the right hand side of the equation becomes 1 (as the exponent is 0) (every multinomial vector is raised to 0) so all of the vectors

$$p_k$$

are equiprobable as seen in the top left of 5 as  $\alpha$  gets larger, the probability distribution concentrates around the mean.

$$P(\mathbf{p} | \alpha \mathbf{m}) = \frac{\Gamma(\sum_k \alpha m_k)}{\prod_k \Gamma(\alpha m_k)} \prod_k p_k^{\alpha m_k - 1}$$



Figure 4: The dirichlet distribution

If



$$\alpha < n$$



where  $n$  is the number of outcomes in the multinomial distribution. When this happens, the probability mass is pushed to the edges of the simplex (bottom right). This means that there isn't too many outcomes with high probability. The advantage of this approach is that not all topics have high probability, meaning some documents can have 1 or 2 substantial topics. With the Matrix Factorisation approach each document had a value for each topic, so a document is assumed to be made up of all the topics in the analysis at varying proportions. LDA has the ability to assume a document is made up a small number of topics which

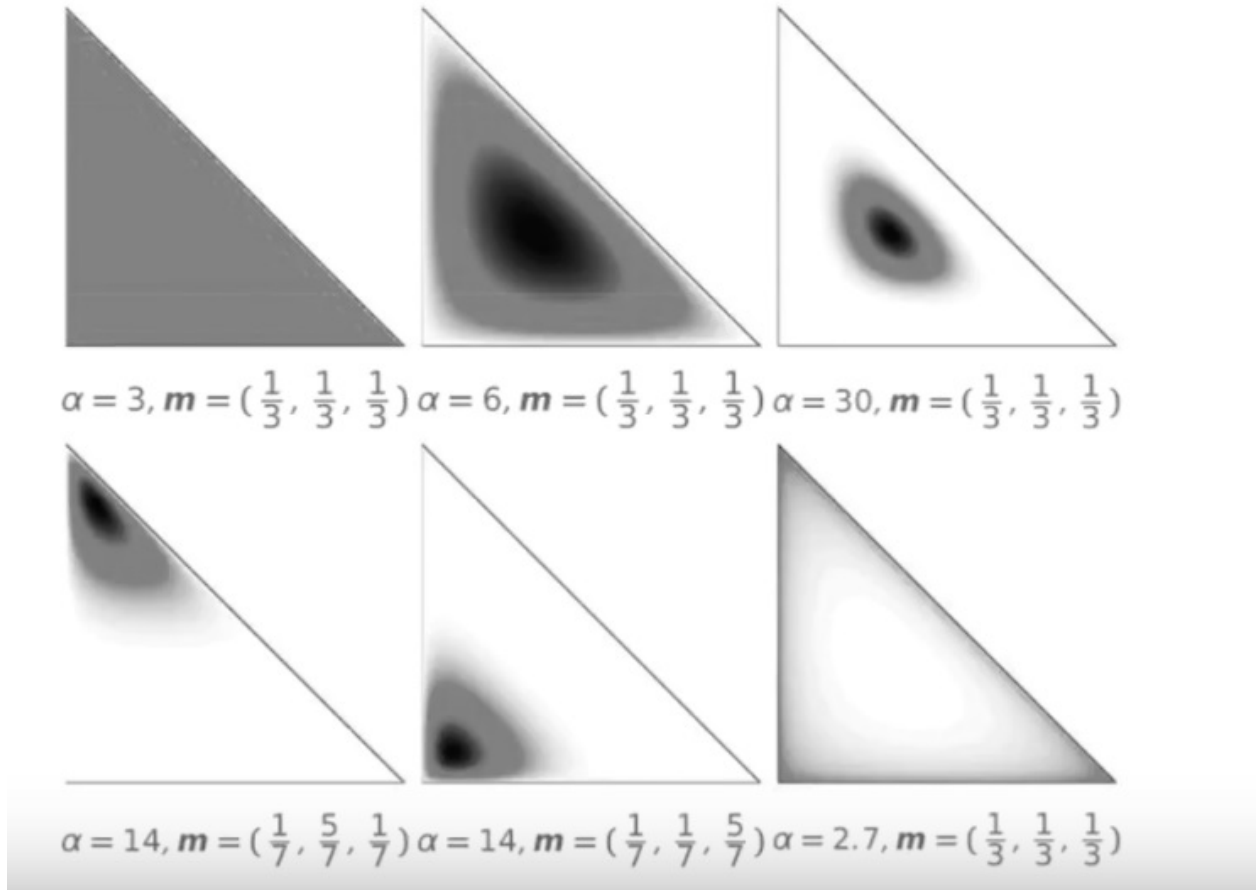


Figure 5: Figure caption4



is a natural assumption as often pieces of texts only deal with one or two topics, for example, Technology and Science.

Using the Dirichlet prior is useful because it allows the model to estimate that the documents are made up of a few topics with one or two topics being stronger than the others. This is intuitive to how real life documents are made and aids interpretability.

### 3.3.1 Dirichlet in detail

With LDA, there is a Dirichlet prior on  $\theta$ . This is a conjugate prior for the multinomial. The probability density of a  $T$ , the number of topics, dimensional Dirichlet distribution over the multinomial distribution  $p = (p_1, \dots, p_T)$  and is given by:

$$Dir(\alpha_1, \dots, \alpha_T) = \frac{\Gamma(\sum_j \alpha_j)}{\prod_j \Gamma(\alpha_j)} \prod_{j=1}^T p_j^{\alpha_j - 1} \quad (2)$$

The parameters of this distribution are specified by  $\alpha_1, \dots, \alpha_T$ . Each hyperparameter  $\alpha_j$  can be interpreted as a prior observation count for the number of times topic  $j$  is sampled in a document before having observed any actual words from that document.

The Dirichlet prior on the topic distribution  $\theta$  results in a smoothed topic distribution depending on the  $\alpha$  parameter. This parameter controls the number of topics for each documents or in other words it controls the probability distribution of topics for each document.

Sampling from the Dirichlet distribution means getting a multinomial distribution for the topics for that document. The  $\alpha$  is a hyperparameter, that is a parameter of parameters, that dictates what the multinomial distribution of topics looks like for each document.

The benefit of the Dirichlet distribution is that for  $\alpha < 1$  the multinomial distribution is often represented by a few topics, maybe 1 to 3 topics. If  $\alpha > 1$  the distribution is quite even - that is the distribution is represented by many topics.

It is more natural to assume a document only contains a few topics, just like a document in real life. For that reason an  $\alpha < 1$  is used in topic modeling.

A Dirichlet distribution prior is also placed on  $\phi$ , the multinomial probability distribution of words of each of the topics, using the parameter  $\beta$ . It can be interpreted as the prior observation count on the number of times words are sampled from a topic before any word from the corpus is observed. Just like the  $\alpha$  parameter, it controls the word distribution for each topic, either making the probability distribution uniform or having only a few words have high probabilities for each topic.

In summary:

The Dirichlet distribution has two parameters:

$$\alpha$$

- controls how different the probabilities will be for words in the topics.

$$\beta$$

- controls how different the probabilities will be for topics in the document.

### Plate notation

Figure 6 shows a way to graphically represent the LDA model. Shaded and unshaded areas indicate observed and latent variables respectively.  $\phi$ , the distribution of words in the topics,  $\theta$ , the distribution of topics in each document and  $z$ , the assignment of word tokens to topics are the three latent variables we aim to infer.

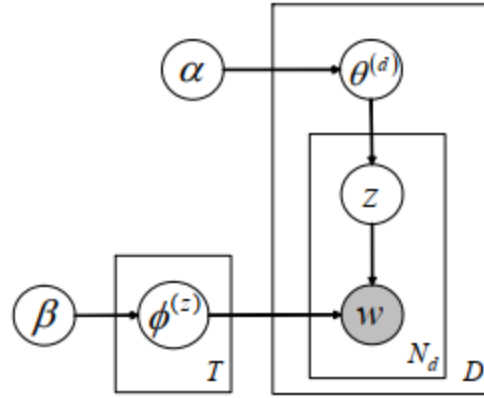


Figure 6: Box diagram

$\alpha$  and  $\beta$  are the hyperparameters that we treat as constants in the model. Arrows represent conditional dependencies between the variables and the plates, the boxes in the figure, refer to repetitions of sampling with the variable in the right corner referring to the number of samples.

For example, the inner plate that has the bubbles  $z$  and  $w$  illustrated the repeated sampling of topics and words until  $N_d$  words have been generated for document  $d$ .

The plate surrounding  $\theta^{(d)}$  shows the repeated sampling of a distribution of topics for each document  $d$  for a total of  $D$  documents.

The plate surrounding  $\phi^{(z)}$  shows the repeated sampling of word distributions for each topic  $z$  until  $T$  documents have been generated.

Here is a labeled version of Figure 6

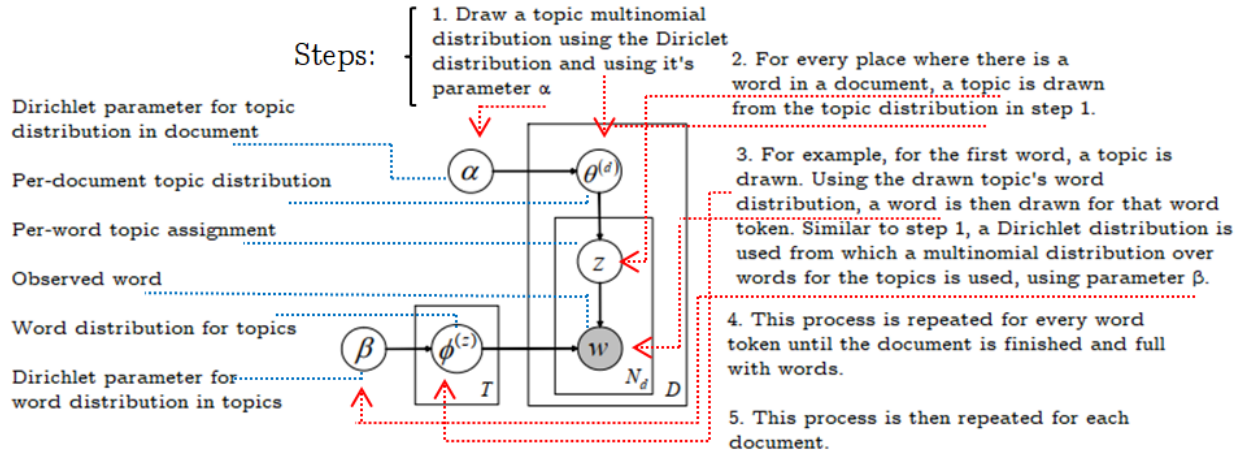


Figure 7: Box diagram labeled

The goal of LDA is to estimate  $\theta$  and  $\phi$ . These are latent variables that aren't observed that we try to fit. We determine these latent variable by using the observed words we have in the corpus, so we reverse engineer to get the parameters that make those observed words. LDA aims to find the parameters that maximise the likelihood of the words we observe in the corpus.

Unlike statistical models like OLS, there is no analytic solution with LDA. The best solution can be approximated in an iterative way, using steps and getting closer to the best solution. One way to do this is with

Gibbs Sampling.

### 3.4 Gibbs sampling

Here is the intuition behind Gibbs sampling. Assume we know the topics of all the words except for the word token of interest. For this word token of interest, a topic must be chosen from the topic distribution for the document. The model prefers to choose a topic that is already present in the document. At the same time, the model also wants to pick a topic that the word already occurs in. So there are 2 probability distributions: one that favors the topics already present in the document and one that favors the topics that this word is already present in.

Taking these two probability distributions together, a joint probability distribution is gained and will give you the chance of picking a topic for this one missing word.

Then, using the iterative Gibbs Sampling in this way:

1. We start with random sampling - we assign every word in every document to one topic.
2. We then iterate over all the words and all the documents. For each word in each document, we compute the proportion of topics in that document (disregarding the word itself).
3. We compute the proportion of topics for that word (disregarding the word itself). I.e what are the topics that the word occurs in and at what frequency?
4. We then multiple these two distributions to get a joint probability distribution.
5. We choose a new topic from that probability distribution and then update the proportions.
6. We repeat this step, going to the next word, and eventually to the next document, until we have updated every word in every document.
7. Initially we started with completely random distributions, but now after these steps we have a slightly less random distribution because the this method, for the words, tends t choose topics that are consistent with that document.
8. Each time these steps are run we get closer to a stable solution, and these steps are run until it converges to a stable solution.

#### 3.4.1 Gibbs Sampling in detail

Gibbs sampling is used. This is a form of the Markov Chain Monte Carlo sampling method which samples values from high-dimensional distributions. Gibbs sampling simulates a complex distribution by sampling over marginal distributions given a specific value of the other distribution. This is done sequentially until the sampled values approximate the distribution we are trying to get.

The Gibbs procedure considers each word token in a document and estimates a probability of assigning the current word token to each topic, conditioned on the topic assignments on the topic assignments to all the other word tokens. From this conditional distribution, a topic is sampled and then is the new topic assignment for this word token. This conditional distribution can be written as:

$$P(z_i = j \mid z_{-i}, w_i, d_i, \cdot) \quad (3)$$

$z_i = j$  represents the topic assignment of token  $i$  to topic  $j$ ,  $z_{-1}$  refers to the topic assignments of all the other word tokens and the dot refers to all other known or observed information as such as all the other word and document indices  $w_{-i}$  and  $d_{-i}$  and hyperparamters  $\alpha$  and  $\beta$ .

This probability is proportional to

$$P(z_i = j \mid z_{-i}, w_i, d_i, \cdot) \propto \frac{C_{w_{ij}}^{WT} + \beta}{\sum_{w=1}^W C_{w_j}^{WT} + W\beta} \frac{C_{d_{ij}}^{DT} + \alpha}{\sum_{t=1}^T C_{d_{ij}}^{DT} + T\alpha} \quad (4)$$

Where:

- $C^{WT}$  and  $C^{DT}$  are matrices of counts
- $C_{wj}^{WT}$  contains the number of times word  $w$  is assigned to  $j$ , not including the current iteration of  $i$
- $C_{dj}^{DT}$  contains the number of times topic  $j$  is assigned to some word token in document  $d$ , not including the current iteration  $i$ .

Note that equation (4) isn't the actual probability of assigning a word token to topic  $j$  but it is proportional to the right hand side. The actual probability is produced by dividing the equation by the sum of all topics  $T$ .

The left part of the right hand side of the equation can be seen as the probability of word  $w$  under topic  $j$ . The right part is the probability that topic  $j$  has under the current topic distribution for document  $d$ .

Because on the top of the left part  $C_{wj}^{WT}$  contains the number of times word  $w$  is assigned to  $j$ , it means that when many tokens of a word have been assigned to topic  $j$  across the documents, it will increase the probability of assigning any particular token of that word to topic  $j$ .

At the same time, looking at the right part of the right hand side, if topic  $j$  has been used multiple times in one document, this means a higher  $C_{dj}^{DT}$ , the number of times topic  $j$  is assigned to some word token in document  $d$ , it will increase the probability that any word from that document will be assigned to topic  $j$ .

As a result, words are assigned to topics depending on how likely the word is for the topic and also how dominant that topic is in a document.

1. Assign each word token to a random topic.
2. Decrease the count matrices  $C^{WT}$  and  $C^{DT}$  by one for the entries that correspond to the current topic assignment.
3. A new topic is sampled from the distribution in Equation 4 and the count matrices  $C^{WT}$  and  $C^{DT}$  are incremented with the new topic assignment.
4. Repeat these steps for all  $N$  word tokens in the corpus, achieving one Gibbs sample.
5. Repeat steps 1-4 until a stable solution converges - that is the posterior distribution over topic assignments are approximated as best as possible.

### Obtaining the topic and word distributions.

Recall that:

- $\phi^{(j)} = P(z = j)$ , a multinomial distribution over words for topic  $j$
- $\theta^{(d)} = P(z)$ , a multinomial distribution over topics for document  $d$

These can be obtained as follows:

$$\phi_i'^{(j)} = \frac{C_{ij}^{WT} + \beta}{\sum_{k=1}^W C_{kj}^{WT} + W\beta} \quad (5)$$

and

$$\theta_j'^{(d)} = \frac{C_{dj}^{DT} + \alpha}{\sum_{k=1}^T C_{dk}^{DT} + T\alpha} \quad (6)$$

## 3.5 Benefits of LDA

LDA adds a Dirichlet prior on top of the data generating process, meaning NMF qualitatively leads to worse mixtures. It fixes values for the probability vectors of the multinomials, whereas LDA allows the topics and words themselves to vary.

Thus, in cases where we believe that the topic probabilities should remain fixed per document (oftentimes unlikely)—or in small data settings in which the additional variability coming from the hyperpriors is too much—NMF performs better. Otherwise, LDA is more suitable to use.

## References