Literature Review on Topic Modeling

Giulia Paggini ID: 3144652

June-September 2022

1 Introduction

With the rapid accumulation of information over the internet, a manual exploration of the text data is often impractical and such a thing of the past. In recent years, several machine learning methods for data analysis have become very much popular among researchers, especially topic models from Natural Language Processing have received lots of attention for information retrieval (IR), in particular techniques for text summarization and document classification.

In these overview below, we will cover the basics of topic modeling, with different mathematical formulations of the same principle, moving on from its historical origins until its latest applications in a variety of fields.

2 What is topic modeling?

Topic models, in a nutshell, are a type of statistical language models used for uncovering latent structure in a collection of texts. Such a topic detection can be done both online and offline modes: in the first case, the purpose is to track the topic changes over time, in the latter, instead, the documents in a corpus are treated as a batch and analyzed one at a time. More intuitively, topic modeling is nothing but a task of dimensionality reduction, unsupervised learning and tagging at the same time.

There are several existing algorithms you can use to perform the topic modeling, the most widespread are: Latent Semantic Analysis LSA/LSI), Probabilistic Latent Semantic Analysis ($pLSA$), and Latent Dirichlet Allocation (LDA).

But, first of all, let's make a step a back to introduce a bit of notation before diving into the fundamentals. We will refer multiple times to some entities like "words", "documents" and "corpora" and it is worth clarifying that:
- a *word* is the simplest unit of text data, defined as an indexed item $\{1, ..., V\}$ from a vocabulary; the most common representation is to encode words as a

unit-basis vectors, that have one component equal to one and all the others set to zero;
- a *document* is a sequence of N words;
- a *corpus* is a collection of M documents denoted by $D = \{w_1, w_2, ..., w_M\}$.

3 Foundations of modern topic modeling

In the 1980's the field of Information Retrieval produced a very well-known scheme called term frequency–inverse document frequency ($tf - idf$) for applying a numerical statistic to a word that would be representative of its relevance within a document in a collection or corpus (Salton and McGill, 1983).

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

The formula can be broke down into two metrics: the term frequency (TF), namely how many times a word appears in a document, and the inverse document frequency (IDF) of the word across a set of documents, so how common or rare a word is in the entire document set. A very common word will have a score around 0, otherwise will approach 1.

Note that it could happen for a term to not appear in the corpus at all, which can result in a divide-by-zero error. One way to handle this is to take the existing count and add 1, thus making the denominator $(1 + df)$.

Here below I provide you with a simple example I prepared using Python, so that we can directly observe how the library Scikit-Learn performs and masters these issues. First of all, I have imported all the required modules needed, then I have created a small dataset (for the sake of simplicity, I decided to include just a couple of short phrases) then I have encoded it using $tfIdVectorizer$ and finally displayed the results using the method .head().I could have also used other classical tools for data visualization, but this was the most immediate one. On purpose, I did not include all the words in the printed output as their

relative score was zero, therefore not that relevant.

```
import pandas as pd
from sklearn.feature_extraction.text import
TfidfTransformer, TfidfVectorizer, CountVectorizer

dataset = [
    "I enjoy reading books about Machine Learning
and its applications",
    "I would enjoy a more advanced course in ML",
    "It is going to rain today",
    "Today I am not going outside",
    "I was reading in the library”]

tfIdfVectorizer=TfidfVectorizer(use_idf=True)
tfIdf = tfIdfVectorizer.fit_transform(dataset)
df = pd.DataFrame(tfIdf[0].T.todense(),
index=tfIdfVectorizer.get_feature_names(),
columns=["TF-IDF"])
df = df.sort_values('TF-IDF', ascending=False)
print (df.head(20))
```

```
              TF-IDF
about       0.347067
books       0.347067
machine     0.347067
its         0.347067
learning    0.347067
applications  0.347067
and         0.347067
enjoy       0.280011
reading     0.280011
more        0.000000
was         0.000000
today       0.000000
to          0.000000
the         0.000000
rain        0.000000
outside     0.000000
not         0.000000
ml          0.000000
course      0.000000
am          0.000000
```

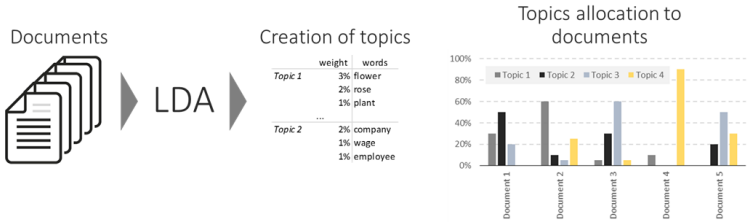
$Tf - idf$ was conceived for document search but it proved to be very useful for extracting keywords from text. However, $tf - idf$ has some limitations. The biggest advantages of $tf - idf$ come from how simple to use it is. It is easy to calculate, computationally cheap, useful when comparing two documents, and it is a natural starting point for similarity calculations (for example via vectorization and cosine similarity). On the other hand, this model only considers the importance of the words due to how it weighs them but not their contexts, nor their order (BoW).

Furthermore,$tf - idf$ could suffer from the curse of dimensionality and so of memory-inefficiency: this is not always the case but it may be an issue in clustering scenarios as the number of documents increases a lot, or more in general when comparing across several documents.

A further step towards $tf-idf$ is represented by LSI (*Latent Semantic Indexing*), which was indeed considered the basic technique for thematic discovery from the text archives. Essentially, LSI transforms the original dataset in different spaces such that documents and words about the same concept can be mapped one to another. Words and documents have many to many relationships. is built using a Singular Value Decomposition (SVD), therefore in case of large text data, the same word can have multiple meanings and imposes a limitation on the LSI range of applicability. The latter issue was overcome later by a slight different model called pLSI (*Probabilistic Latent Semantic Indexing*) where words and topics are mapped together according to some specific probability, scoring, of course, better performances compared to LSI. Still, as LSI, pLSI was not a fully generative model, or in other terms, it was not able to explain unforeseen documents that are not part of the given collection.

4 Latent Dirichlet Allocation

Blei et al. first introduced Latent Dirichlet Allocation, a hierarchical, probabilistic and unsupervised generative model that takes documents as inputs and finds topics as output. More specifically, a topic is nothing but a weighted list of words. Intuitively, the three main parameters of the model are: the number of topics, the number of words per topic and the number of topics per document.

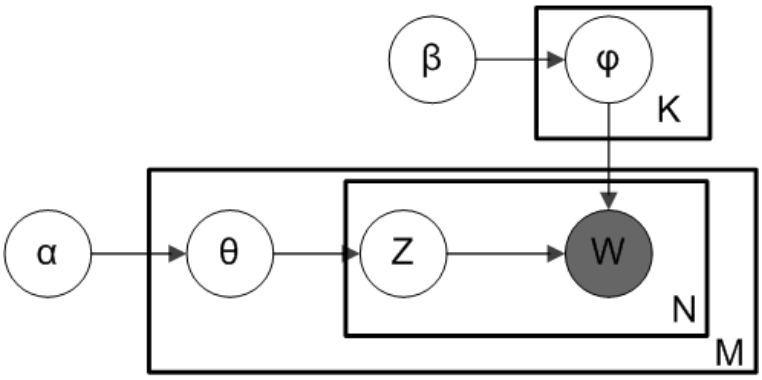


Behind the generative process, some assumptions are needed:
- a document is a bag of words, namely the order does not count;
- each document is a mix of topics, each of them appears in a certain proportion in the document;
- each topic is a mix of words.

Within this framework, it is clear that the proportion of all topics sum up to 1 in each document. The process instead consists of three levels that involve the whole corpus, the documents, and the terms of each document. The algorithm

first samples, for each document, a distribution over collection topics from a Dirichlet distribution. Then, it picks a single topic for each of a document's terms according to this distribution. Finally, each term is then sampled from a Multinomial distribution over terms specific to the sampled topic.

The complete LDA generation process can be better understood graphically through its plane notation here below, with the following variables names:
- M denotes the number of documents;
- N is number of words in a given document (document i has N_i words);
- α is the parameter of the Dirichlet prior on the per-document topic distributions;
- β is the parameter of the Dirichlet prior on the per-topic word distribution;
- θ_i is the topic distribution for document i;
- φ_k is the word distribution for topic k;
- z_{ij} is the topic for the j-th word in document i;
- w_{ij} is the specific word.



The theoretical foundation of LDA rely on exploiting the concepts of exchangeability with the de Finetti theorem (1990). Exchangeability is a major simplifying assumption of text processing that allows for computationally-efficient methods. In fact, both LSI and pLSI and based on the fundamental probability assumption described by the "bag-of-words" method where the order of words in a document can be ignored.
This assumption of exchangeability applies also to the treatment of documents, where one can assume that the specific order of documents in a corpus is not an important consideration.

According to de Finetti's theorem, exchangeable observations are conditionally independent relative to a latent variable, therefore a probability can be assigned to the latent variable. Moreover, any collection of exchangeable random variables has a representation as a "mixture" distribution, specifically a mixture of

sequences of independent and identically distributed (*i.i.d.*) Bernoulli random variables. The key implication of such a model is the possibility of representing the presence of sub-populations within an overall population without requiring to identify the sub-populations within the dataset. In short, by using De Finetti's theorem, it is possible to capture significant intra-document statistical structure via the mixture distribution.

The procedure is simply done as follows:

Draw each topic parameter $\beta_k \sim \text{Dirichlet}(\varphi)$ for $k \in \{1, ..., K\}$
For each document:

- 1) Choose the topic distribution $\theta_i \sim \text{Dirichlet}(\alpha)$
- 2) For each of the N words $w_{i,j}$:

  a) Choose a topics $z_{i,j} \sim \text{Multinomial}(\theta_i)$

  b) Choose a word $w_{i,j} \sim \text{Multinomial}(\beta_k)$

The probability of a corpus is the result of the product of the marginal probabilities of single documents, and the marginal distribution for a single document is given by integrating over θ and summing over z topics.

To better understand the formula described above, we need to briefly recall the Bayes' Theorem which states that, if the prior is distributed according to Dirichlet(α, β) and the likelihood is Multinomial distributed$(z_{i,j}, w_{i,j})$, then the posterior can be computed and will be Dirichlet distributed too. Tough, the posterior distribution cannot be computed directly but approximated by several algorithms, for example Markov Chain MC (e.g. Gibbs sampling), LaPlace approximation, etc.From a more practical standpoint, LDA can be easily executed thanks to several Python libraries, like Sci-Kit Learn or Gensim, actually one of the most popular.

When implementing Latent Dirichlet Allocation algorithm, it is very important to fine-tune the hyperparameters we are dealing with, namely the number of topics k, the number of features, α and β prior parameters. In principle there are many other parameters that can be tuned during the process, but we will not focus on them in this review.

- The number of topics k depends on the characteristics of the dataset and its size, in general the larger the dataset, the higher the number of topics. A feasible approach to find out the optimal number of topics is to compute the topic coherence scores over a range ot topic numbers and

plot the trends. As it may get time consuming, one could simply plug a starting value and empirically adjust it.

- A similar approach could be used for the number of features, namely setting a fixed size for the vocabulary: setting it to 10,000 is mostly fine for not too complex models. On the other hand, by increasing this number, you may end up with a diminishing return on clustering accuracy, as many words will result in a small percentage (see previous example with td-idf vectorization).

- In most of the LDA models, α and β parameters, that represent document-topic density and topic-word density respectively, come from a symmetric distribution, so that each topic is evenly distributed throughout a document. The α parameter specifies our beliefs about topics uniformity and sparsity, an higher of α implies more topics in the documents, conversely, a low value assumes that a document will contain a mixture of just a few or a single topic. The β parameter will specify prior beliefs about word sparsity and uniformity within topics, high values means that topics contains most of the words in the fixed-size vocabulary, the viceversa is true for low values of β.

5 Benefits and limitations of LDA

Latent Dirichlet Allocation was most commonly employed for strategic business optimization. Nowadays, large corporations, but also small firms, are interested in exploiting the huge amount of data available as much as possible in order to boost their visibility on the marketplace and performance. LDA-models has been used on online text publications to uncover the key personality traits of consumers,as well as their needs or complaints, in order to develop new tailored and appealing products.

Nevertheless, LDA has been largely criticised for a variety of reasons. First of all, LDA is not able to scale due to the linearity of the technique it is built on, whereas instead other algorithms, like pLSI, are able to deal with this issue. Another pitfall of this model is the fundamental assumption of exchangeability. In some situations, such a principle could be very restrictive, in particular when dealing with topics that evolve over time.

In addition, LDA-based models are torn down for neglecting co-occurrence relations across the documents which are of interest. This leads to an incomplete detection of information and an inability to discover latent co-occurrence relations via the context or other bridge terms.
Lastly, LDA is not best suited for short, user-generated text, that is why Hajjem and Latiri (2017) proposed an hybrid model to overcome problems of efficiency and noise fitting. In particular, apart from these two, other limitations apply:

- abbreviations and slang: micro-blogging involves a flexible language, that is unanimously considered more demanding than traditional text for algorithmic analysis. The challenge is mainly due to sarcasm, imagery, metaphors and other traits typically subject matter of what is called sentiment analysis;

- need for extensive data pre-processing;

- fixed value for K, the number of topics is fixed and must be known ahead of time;

- static model;

- lack of context, due to the few words available.

To solve some of the limitations of LDA mentioned above, researchers came up with new models that learn vector representation of words. By learning in this way, they have a more effective classification performance on short-form text. For instance, Yu and Qiu proposed a hybrid model, where the user-LDA topic model is extended with the Dirichlet multinomial mixture and a word vector tool, delivering an optimal performance, when compared to other hybrid models or the LDA model alone on micro-blog textual data.

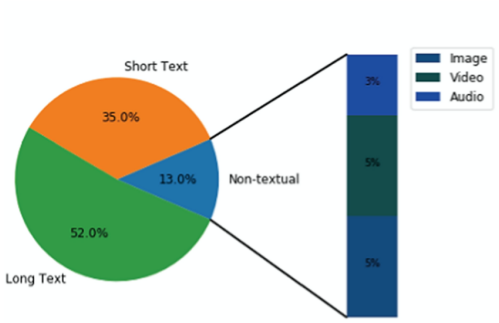
A similar approach could be applied to Twitter data. This is the hierarchical latent Dirichlet allocation ($hLDA$), which aims to automatically mine the hierarchical dimension of tweets' topics by using word2vec (i.e. a vector representation technique). By doing so, it extracts semantic relationships of words in the data to obtain a more effective dimension.

Other approaches like the Non-negative matrix factorization (NMF) model have also been recognized to perform better than LDA on short text under similar configurations.

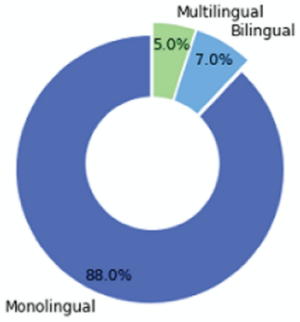
6 Conclusions

In this brief review, we developed a concise explanation of what is topic modeling and its various steps towards the years. Although LDA is extensively employed for several purposes, the preprocessing such as stopdwords elimination, lemma-

tization and stemming still need to be developed for different languages.

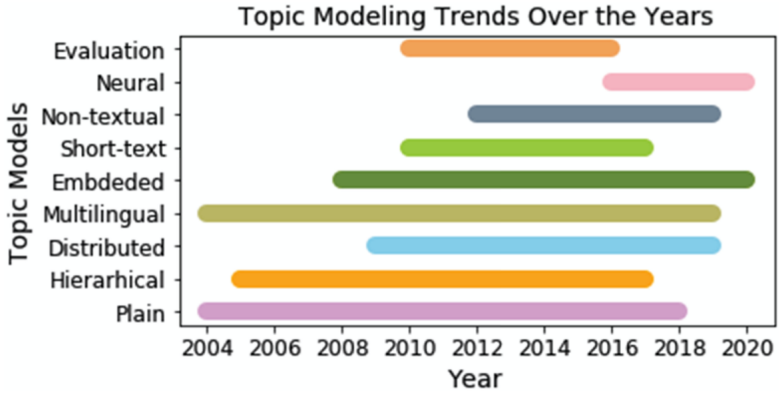


(a) Proportion of textual and non-textual datasets



(b) Topic Modelings in monolingual and multilingual setup

Another remark that is worth to mention is proposed in the Uttam Chauhan and Apurva Shah. 2021 article about the main interest of topic modelers, in particular it was found out that textual corpus was more attractive than non-textual corpus, likewise, monolingual topic modeling has received major attention against bilingual and polylingual work, as showed in the two graphs above. Besides all the improvements in this field, the topic modeling trends seem to have evolved a lot in the past years, from Plain to Embedded Topic modeling, from Short text to Neural models, which indeed are the most promising for the next future.



7 References

- Uttam Chauhan and Apurva Shah. 2021. Topic Modeling Using Latent Dirichlet allocation: A Survey

- Blei, D. M., Griffiths, T. L., and Jordan, M. I. 2010. The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies

- Blei, D. M., Ng, A. I., and Jordan, M. I. Journal of Machine Learning Research 3 (2003) 993-1022

- https://towardsdatascience.com/topic-modelling-f51e5ebfb40a
https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation