Python with LDA

Using LDA (Latent Dirichlet Allocation) for topics extraction from a corpus of documents

Félix Revert Follow
Dec 17, 2018 · 7 min read

A recurring subject in NLP is to understand large corpus of texts through topics extraction. Whether you analyze users' online reviews, products' descriptions, or text entered in search bars, understanding key topics will always come in handy.

Popular picture used in literature to explain LDA

Before going into the LDA method, let me remind you that not reinventing the wheel and going for the quick solution is usually the best start. Several providers have great API for topic extraction (and it is free up to a certain number of calls): Google, Microsoft, MeaningCloud… I tried all of the three and all work very well.

However, if your data is highly specific, and no generic topic can represent it, then you will have to go for a more personalized approach. This article focuses on one of these approaches: **LDA**.

Understanding LDA

Intuition

LDA (*short for Latent Dirichlet Allocation*) is an unsupervised machine-learning model that takes documents as input and finds topics as output. The model also says in what percentage each document talks about each topic.

A topic is represented as a weighted list of words. An example of a topic is shown below:

> *flower * 0,2 | rose * 0,15 | plant * 0,09 |…*

Illustration of LDA input/output workflow

There are 3 main parameters of the model:

- the number of topics
- the number of words per topic
- the number of topics per document

In reality, the last two parameters are not exactly designed like this in the algorithm, but I prefer to stick to these simplified versions which are easier to understand.

Implementation

[A dedicated Jupyter notebook is shared at the end]

In this example, I use a dataset of articles taken from BBC's website.

To implement the LDA in Python, I use the package *gensim*.

```
from gensim import corpora, models

# list_of_list_of_tokens = [["a","b","c"], ["d","e","f"]]
# ["a","b","c"] are the tokens of document 1, ["d","e","f"]
dictionary_LDA = corpora.Dictionary(list_of_list_of_tokens)
dictionary_LDA.filter_extremes(no_below=3)
corpus = [dictionary_LDA.doc2bow(list_of_tokens) for list_o
```

A simple implementation of LDA, where we ask the model to create 20 topics

The parameters shown previously are:

1. the number of topics is equal to **num_topics**
2. the *[distribution of the]* number of words per topic is handled by **eta**
3. the *[distribution of the]* number of topics per document is handled by **alpha**

To print topics found, use the following:

```
for i,topic in lda_model.show_topics(formatted=True, num_top
    print(str(i)+": "+ topic)
    print()
```

print_topics.py hosted with ❤ by GitHub view raw

```
for i,topic in lda_model.show_topics(formatted=True, num_top
    print(str(i)+": "+ topic)
```

```
0: 0.024*"base" + 0.018*"data" + 0.015*"security" +
0.015*"show" + 0.015*"plan" + 0.011*"part" +
0.010*"activity" + 0.010*"road" + 0.008*"afghanistan" +
0.008*"track" + 0.007*"former" + 0.007*"add" +
0.007*"around_world" + 0.007*"university" +
0.007*"building" + 0.006*"mobile_phone" + 0.006*"point" +
0.006*"new" + 0.006*"exercise" + 0.006*"open"

1: 0.014*"woman" + 0.010*"child" + 0.010*"tunnel" +
0.007*"law" + 0.007*"customer" + 0.007*"continue" +
0.006*"india" + 0.006*"hospital" + 0.006*"live" +
0.006*"public" + 0.006*"video" + 0.005*"couple" +
0.005*"place" + 0.005*"people" + 0.005*"another" +
0.005*"case" + 0.005*"government" + 0.005*"health" +
0.005*"part" + 0.005*"underground"

2: 0.011*"government" + 0.008*"become" + 0.008*"call" +
0.007*"report" + 0.007*"northern_mali" + 0.007*"group" +
0.007*"ansar_dine" + 0.007*"tuareg" + 0.007*"could" +
```

```
0.007*"us" + 0.006*"journalist" + 0.006*"really" +
0.006*"story" + 0.006*"post" + 0.006*"islamist" +
0.005*"data" + 0.005*"news" + 0.005*"new" + 0.005*"local" +
0.005*"part"
```

[the first 3 topics are shown with their first 20 most relevant words]

Topic 0 seems to be about military and war.

Topic 1 about health in India, involving women and children.

Topic 2 about Islamists in Northern Mali.

To print the % of topics a document is about, do the following:

```
lda_model[corpus[0]] # corpus[0] means the first document.
```

print_topic_allocation.py hosted with ❤️ by GitHub view raw


```
lda_model[corpus[0]] # corpus[0] means the first document.
```

print_topic_allocation.py hosted with ❤️ by GitHub view raw


```
[(14, 0.9983065953654187)]
```

The first document is 99.8% about topic 14.

Predicting topics on an unseen document is also doable, as shown below:

```
new_text = 'How many species of animals are there in Russia
tokens = [stemmer.stem(token) for token in tokenizer.tokeniz
lda_model[dictionary_LDA.doc2bow(tokens)]
```

allocate_topics_on_new_document.py hosted with ❤️ by GitHub view raw

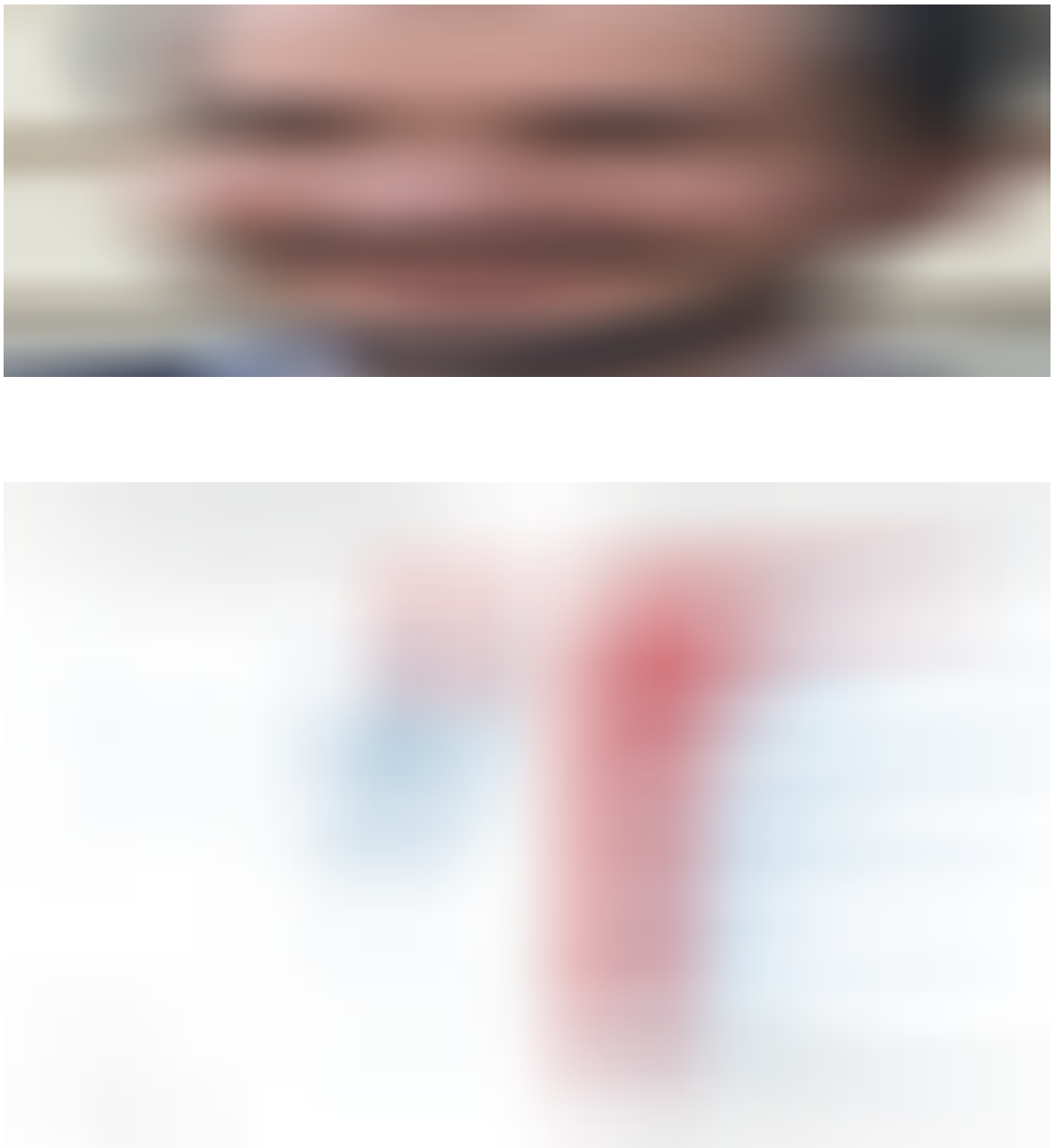

```
new_text = 'How many species of animals are there in Russia
tokens = [stemmer.stem(token) for token in tokenizer.tokeniz
```


```
[(1, 0.5173717951813482), (3, 0.43977106196150995)]
```

This new document talks 52% about topic 1, and 44% about topic 3. Note that 4% could not be labelled as existing topics.

Exploration

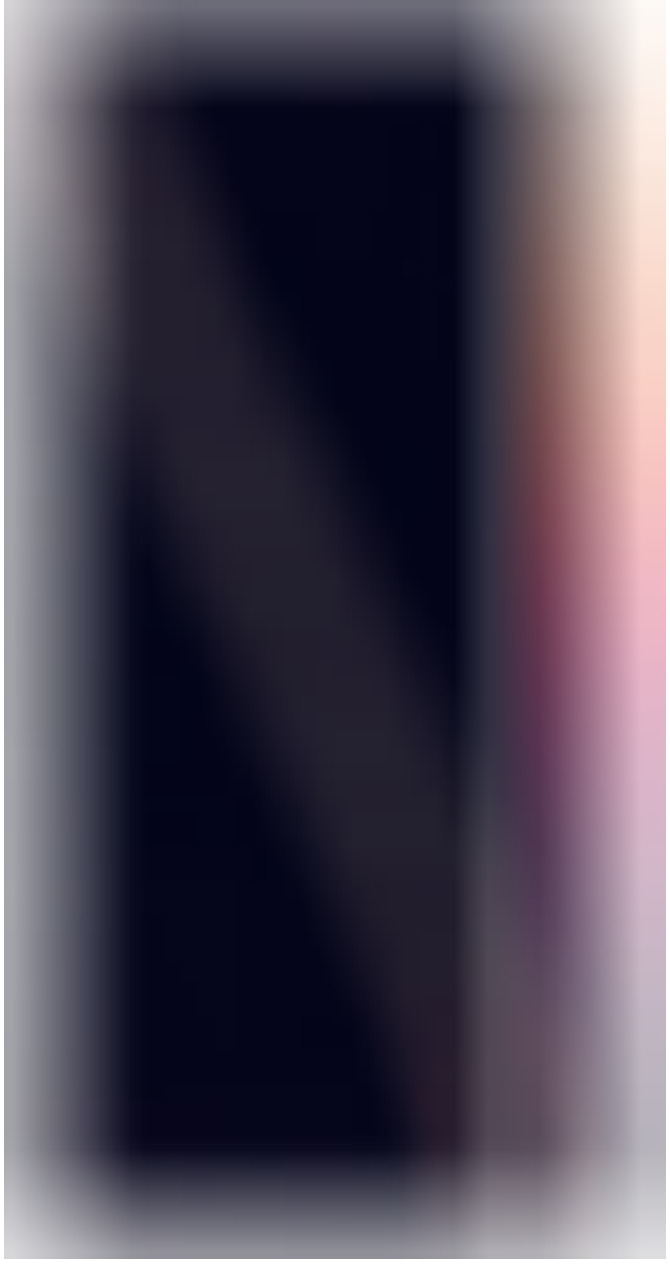
There is a nice way to visualize the LDA model you built using the package *pyLDAvis*:



Output of the pyLDAvis

This visualization allows you to compare topics on two reduced dimensions and observe the distribution of words in topics.

Another nice visualization is to show all the documents according to their major topic in a diagonal format.



Visualization of the proportion of topics in the documents (Documents are rows, topic are columns)

Topic 18 is the most represented topic among documents: 25 documents are mainly about it.

How to successfully implement LDA

LDA is a complex algorithm which is generally perceived as hard to fine-tune and interpret. Indeed, getting relevant results with LDA requires a

strong knowledge of how it works.

Data cleaning

A common thing you will encounter with LDA is that words appear in multiple topics. One way to cope with this is to add these words to your stopwords list.

Another thing is plural and singular forms. I would recommend lemmatizing—or stemming if you cannot lemmatize but having stems in your topics is not easily understandable.

Removing words with digits in them will also clean the words in your topics. Keeping years (2006, 1981) can be relevant if you believe they are meaningful in your topics.

Filtering words that appear in at least 3 (or more) documents is a good way to remove rare words that will not be relevant in topics.

Data preparation

Include bi- and tri-grams to grasp more relevant information.

Another classic preparation step is to use only nouns and verbs using POS tagging (POS: Part-Of-Speech).

Fine-tuning

- Number of topics: try out several numbers of topics to understand which amount makes sense. You actually need to *see* the topics to know if your model makes sense or not. As for K-Means, LDA converges and the model makes sense at a mathematical level, but it does not mean it makes sense at a human level.

- Cleaning your data: adding stop words that are too frequent in your topics and re-running your model is a common step. Keeping only nouns and verbs, removing templates from texts, testing different cleaning methods iteratively will improve your topics. Be prepared to spend some time here.

- Alpha, Eta. If you're not into technical stuff, forget about these. Otherwise, you can tweak alpha and eta to adjust your topics. Start with 'auto', and if the topics are not relevant, try other values. I recommend using low values of Alpha and Eta to have a small number of topics in each document and a small number of relevant words in each topic.

- Increase the number of *passes* to have a better model. 3 or 4 is a good number, but you can go higher.

Assessing results

- Are your topics interpretable?
- Are your topics unique? (two different topics have different words)
- Are your topics exhaustive? (are all your documents well represented by these topics?)

If your model follows these 3 criteria, it looks like a good model :)

Main advantages of LDA

It's fast

Use the *%time* command in Jupyter to verify it. The model is usually fast to run. Of course, it depends on your data. Several factors can slow down the model:

- Long documents
- Large number of documents
- Large vocabulary size (especially if you use n-grams with a large n)

It's intuitive

Modelling topics as weighted lists of words is a simple approximation yet a very intuitive approach if you need to interpret it. No embedding nor hidden dimensions, just bags of words with weights.

It can predict topics for new unseen documents

Once the model has run, it is ready to allocate topics to any document. Of course, if your training dataset is in English and you want to predict the topics of a Chinese document it won't work. But if the new documents have the same structure and should have more or less the same topics, it will work.

Main disadvantages of LDA

Lots of fine-tuning

If LDA is fast to run, it will give you some trouble to get good results with it. That's why knowing in advance how to fine-tune it will really help you.

It needs human interpretation

Topics are found by a machine. A human needs to label them in order to present the results to non-experts people.

You cannot influence topics

Knowing that some of your documents talk about a topic you know, and not finding it in the topics found by LDA will definitely be frustrating. And there's no way to say to the model that some words should belong together. You have to sit and wait for the LDA to give you what you want.

Conclusion

LDA remains one of my favourite model for topics extraction, and I have used it many projects. However, it requires some practice to master it. That's why I made this article so that you can jump over the barrier to entry of using LDA and use it painlessly.

Code: https://github.com/FelixChop/MediumArticles/blob/master/LDA-BBC.ipynb