Short Text Topic Modeling

Intuition and code to understand and implement Topic Modeling on Short Texts like social media ones (Tweets, Reddit's posts…)

Matyas Amrouche
Aug 23, 2019 · 10 min read ★

Photo by Hello I'm Nik GB on Unsplash

Topic Modeling aims to find the topics (or clusters) inside a corpus of texts (like mails or news articles), without knowing those topics at first. Here lies the real power of Topic Modeling, you don't need any labeled or annotated data, only **raw texts**, and from this chaos Topic Modeling algorithms will find the topics your texts are about!

In this post we will describe the intuition and logic behind the most popular approach for Topic Modeling, the **LDA**, and see its **limitation on short texts**. Given this post is about **Short Text Topic Modeling** (**STTM**) we will not dive into the details of LDA. The reader willing to deepen his knowledge of LDA can find great articles and useful resources about LDA here and here.
Then, in a second part, we will present a new approach for **STTM** and finally see in a third part how to easily apply it (fit/predict ✌️) on a toy dataset and evaluate its performance.

The reader already familiar with LDA and Topic Modeling may want to skip the first part and directly go to the second and third ones which present a new approach for **Short Text Topic Modeling** and its **Python coding** 🐍.

. . .

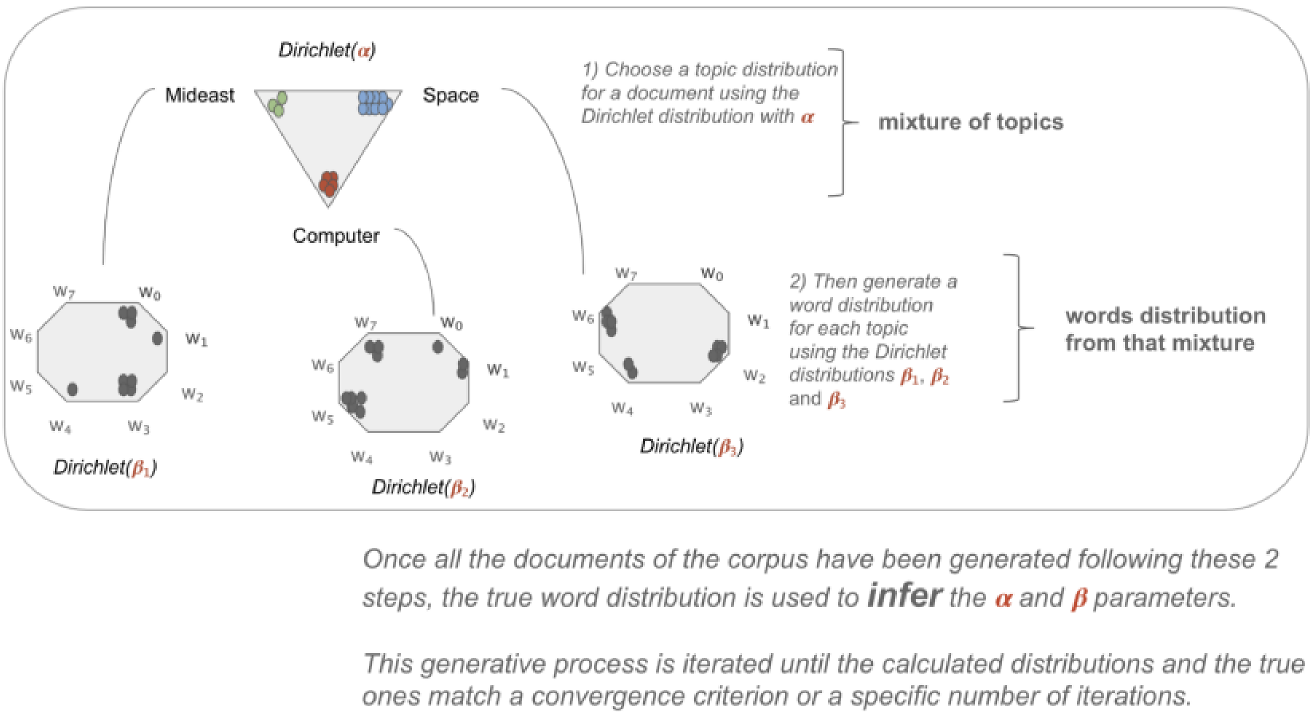
I. Latent Dirichlet Allocation

The most popular Topic Modeling algorithm is **LDA, Latent Dirichlet Allocation**. Let's first unravel this imposing name to have an intuition of what it does.

- **Latent** because the topics are "hidden". We have a bunch of texts and we want the algorithm to put them into clusters that will make sense to us. For example, if our text data come from news content, typically the clusters found might be about Mideast Politics, Computer, Space… but we don't know it yet.
- **Dirichlet** stands for the Dirichlet distribution the model uses as a prior to generate document-topic and word-topic distributions.
- **Allocation** because we want to allocate topics to our texts.

Figure 1 below describes how the LDA steps articulate to find the topics within a corpus of documents.

"A document is generated by sampling a mixture of these topics and then sampling words from that mixture” (Andrew Ng, David Blei and Michael Jordan from the LDA original paper).



1) Choose a topic distribution for a document using the Dirichlet distribution with α

2) Then generate a word distribution for each topic using the Dirichlet distributions β_1, β_2 and β_3

*Once all the documents of the corpus have been generated following these 2 steps, the true word distribution is used to **infer** the α and β parameters.*

This generative process is iterated until the calculated distributions and the true ones match a convergence criterion or a specific number of iterations.

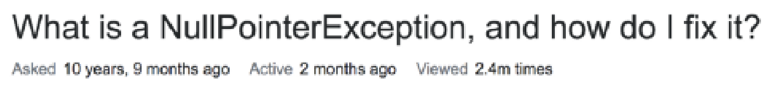
Figure 1: LDA documents generation process using Dirichlet distribution.

NB: In the Figure 1 above, we have set K=3 topics and N=8 words in our vocabulary for illustration ease. We also named these topics Computer, Space and Mideast Politics for illustration ease (rather than calling them topic 1, topic 2 and topic 3). Indeed, it will be our task to understand that the 3 found topics are about Computer, Space and Mideast Politics regarding their content (we will see this part more in depth during the topic attribution of our STTM pipeline in part III).

In short, LDA by using Dirichlet distributions as prior knowledge generates documents made of topics and then update them until they match the ground truth.

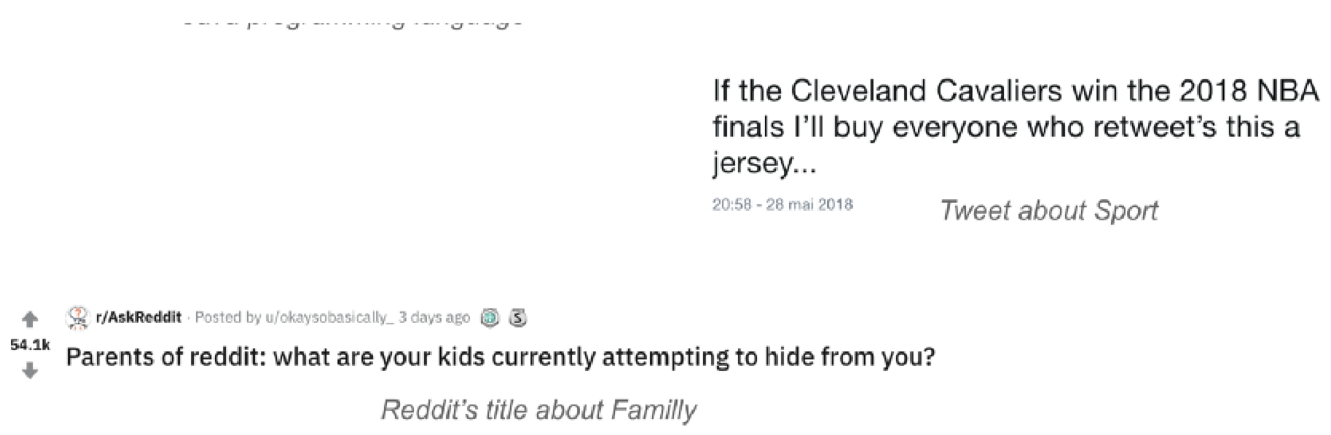
II. Short Text Topic Modeling (STTM)

Despite its great results on medium or large sized texts (>50 words), typically mails and news articles are about this size range, **LDA poorly performs on short texts** like Tweets, Reddit posts or StackOverflow titles’ questions.


What is a NullPointerException, and how do I fix it?

Asked 10 years, 9 months ago Active 2 months ago Viewed 2.4m times

StackOverflow’s title about the Java programming language


If the Cleveland Cavaliers win the 2018 NBA finals I'll buy everyone who retweet's this a jersey...

20:58 - 28 mai 2018

r/AskReddit · Posted by u/okaysobasically_ 3 days ago

54.1k

Parents of reddit: what are your kids currently attempting to hide from you?

Figure 2: Example of short texts and the topic they discuss.

Looking at the short texts examples above on Figure 2, it is evident that the assumption that a text is a mixture of topics (remember first step in Figure 1) is not true anymore. We will now assume that a short text is made from **only one topic**.

The **Gibbs Sampling Dirichlet Mixture Model** (**GSDMM**) is an "altered" LDA algorithm, showing great results on STTM tasks, that makes the initial assumption: **1 topic ↔1 document**. The words within a document are generated using the same unique topic, and not from a mixture of topics as it was in the original LDA.

Before diving into code and practical aspects, let's understand **GSDMM** with an equivalent procedure called the **Movie Group Process** that will help us understand the different steps and process under the hood of STTM, and how to **tune efficiently its hyper-parameters** (we remember alpha and beta from the LDA part).

Imagine a bunch of students in a restaurant, seating randomly at K tables. They are all asked to write their favorite movies on a paper (but it must remain a short list). The objective is to cluster them in such a way that so students within the same group share the same movie interest. To do so, one after another, students must make a new table choice regarding the two following rules:

- Rule 1: Choose a table with more students. This rule improves **completeness**, all students sharing the same movie's interest are assigned to the same table.
- Rule 2: Choose a table where students share similar movie's interest. This rule aims to increase **homogeneity**, we want only members sharing the same movie's interest at a table.

After repeating this process, we expect some tables to disappear and others to grow larger and eventually have clusters of students matching their movie's interest. This is simply what the GSDMM algorithm does!

III. STTM implementation

In this part we will build **full STTM pipeline** from a concrete example using the 20 News Groups dataset from Scikit-learn used for Topic Modeling on texts.

First thing first, we need to download the STTM script from Github into our project folder.

```
cd sttm_project
git clone https://github.com/rwalk/gsdmm.git
```

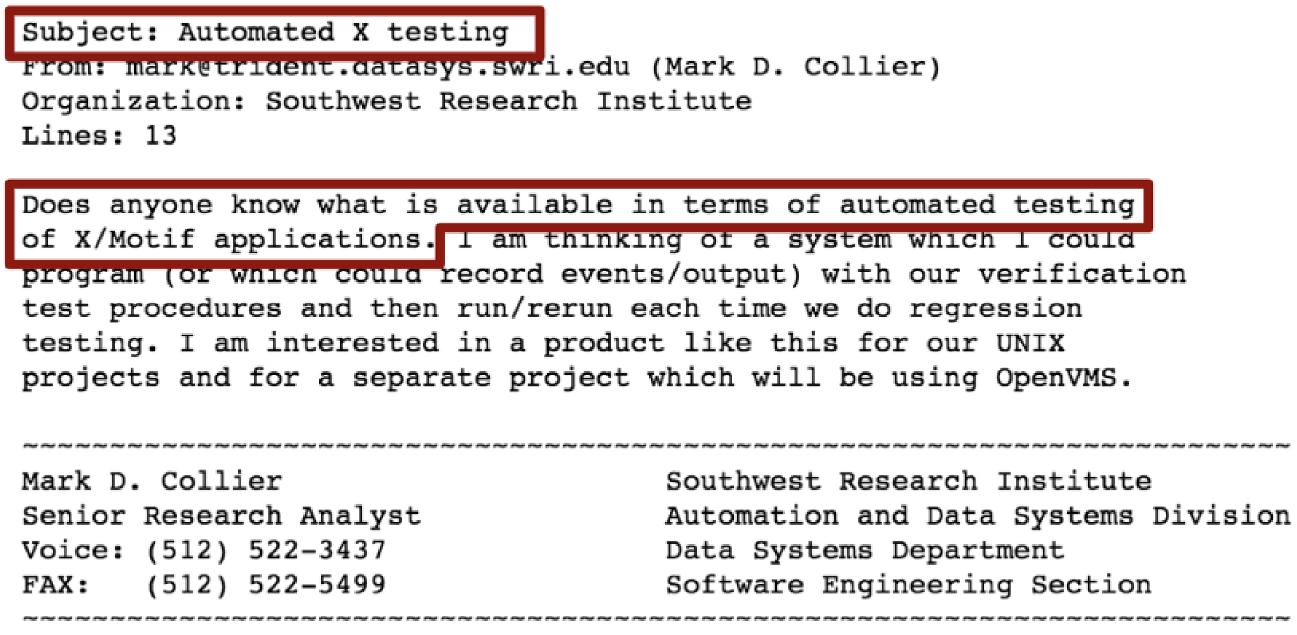
Now, we can start implementing the STTM pipeline (here is a static version of the notebook I used).

```
# Useful libs
from sklearn.datasets import fetch_20newsgroups
import pickle
import pandas as pd
import numpy as np

# STTM lib from Github
from gsdmm import MovieGroupProcess

# Custom python scripts for preprocessing, prediction and
# visualization that I will define more in depth later
from preprocessing import tokenize
from topic_allocation import top_words, topic_attribution
from visualisation import plot_topic_notebook

# Load the 20NewsGroups dataset from sklearn
cats = ['talk.politics.mideast', 'comp.windows.x', 'sci.space']
newsgroups_train = fetch_20newsgroups(subset='train',
categories=cats)
```

However, in this exercise, we will not use the whole content of the news to extrapolate a topic from it, but **only consider the Subject and the first sentence of the news** (see Figure 3 below). Indeed, we need short texts for short texts topic modeling… obviously 🙏

Besides, we will only look at only 3 topics (evenly distributed among the dataset), for illustration ease. These topics are the following:

- Mideast Politics 🌍
- Space 👾
- Windows X 📱



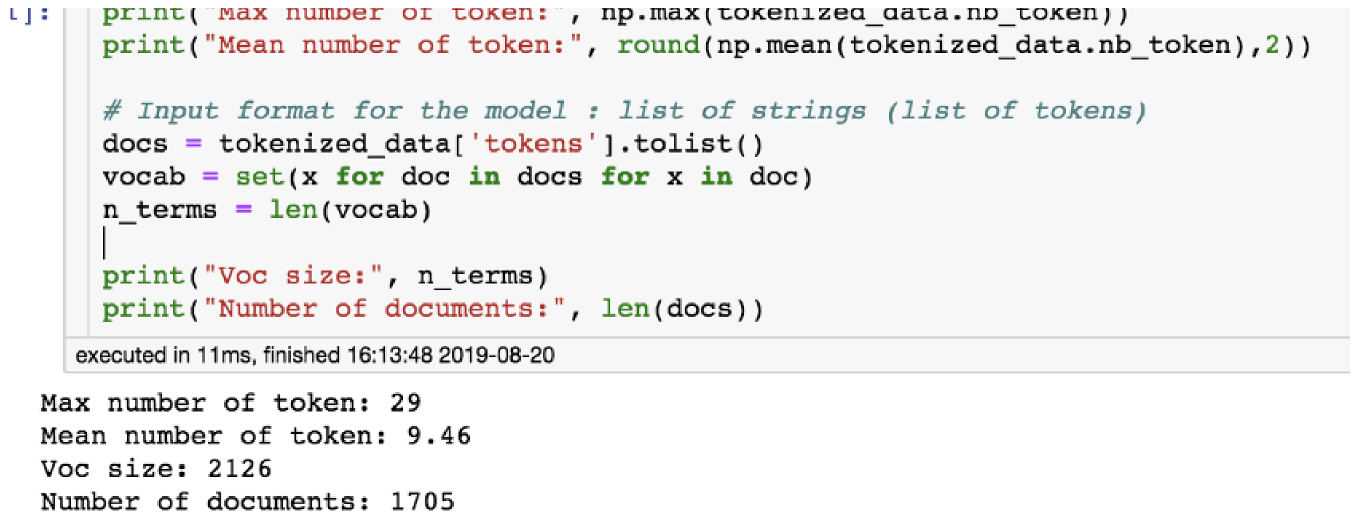
```
Subject: Automated X testing
From: mark@trident.datasys.swri.edu (Mark D. Collier)
Organization: Southwest Research Institute
Lines: 13

Does anyone know what is available in terms of automated testing
of X/Motif applications. I am thinking of a system which I could
program (or which could record events/output) with our verification
test procedures and then run/rerun each time we do regression
testing. I am interested in a product like this for our UNIX
projects and for a separate project which will be using OpenVMS.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Mark D. Collier                     Southwest Research Institute
Senior Research Analyst             Automation and Data Systems Division
Voice: (512) 522-3437               Data Systems Department
FAX:   (512) 522-5499               Software Engineering Section
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 3: Example of a news about the topic: **Windows X**. We concatenate these 2 highlighted sentences together to have a document.

```
# Preprocessing and tokenization
tokenized_data = tokenize(df, form_reduction='stemming',
predict=False)
```

Here are the preprocessing recipe I have followed for this task:

- Tokenization using spaCy tokenizer.
- Removing stop words and 1 character words.
- Stemming (given my empirical experience I have observed that stemming gives better clusters on short text compared to lemmatization) using the nltk library's stemmer.
- Removing empty documents and documents with more than 30 tokens.
- Removing unique token (with a term frequency = 1).



```
[ ]:  print("Max number of token:", np.max(tokenized_data.nb_token))
      print("Mean number of token:", round(np.mean(tokenized_data.nb_token),2))

      # Input format for the model : list of strings (list of tokens)
      docs = tokenized_data['tokens'].tolist()
      vocab = set(x for doc in docs for x in doc)
      n_terms = len(vocab)

      print("Voc size:", n_terms)
      print("Number of documents:", len(docs))
     executed in 11ms, finished 16:13:48 2019-08-20

Max number of token: 29
Mean number of token: 9.46
Voc size: 2126
Number of documents: 1705
```

Figure 4: We do match the Short Text statistics regarding the number of token in our documents, referring to the STTM survey paper p.10.

However, one must keep in mind that **preprocessing is data dependent** and should consider to adapt an other preprocessing approach if a different dataset is used.

Now that our data are cleaned and processed to the proper input format, we are ready to train the model 🚀

```
# Train STTM model

# Init of the Gibbs Sampling Dirichlet Mixture Model algorithm
# K = number of potential topic (which we don't know a priori)
# alpha =
# beta =
# n_iters = number of iterations to
mgp = MovieGroupProcess(K=10, alpha=0.1, beta=0.1, n_iters=30)

vocab = set(x for doc in docs for x in doc)
n_terms = len(vocab)

y = mgp.fit(docs, n_terms)

# Save model
with open('dumps/trained_models/v1.model', "wb") as f:
 pickle.dump(mgp, f)
 f.close()
```

Let's dive under the hood and understand the hyper-parameters machinery of the GSDMM model 🛠:

- **K** = 10. In a real case we are not aware of the exact number of topic so we want to choose a higher value. Theoretically, GSDMM should empty useless clusters and eventually find the exact number of cluster. It will not be the case here, but nothing to worry, we will explain the situation more in depth later.

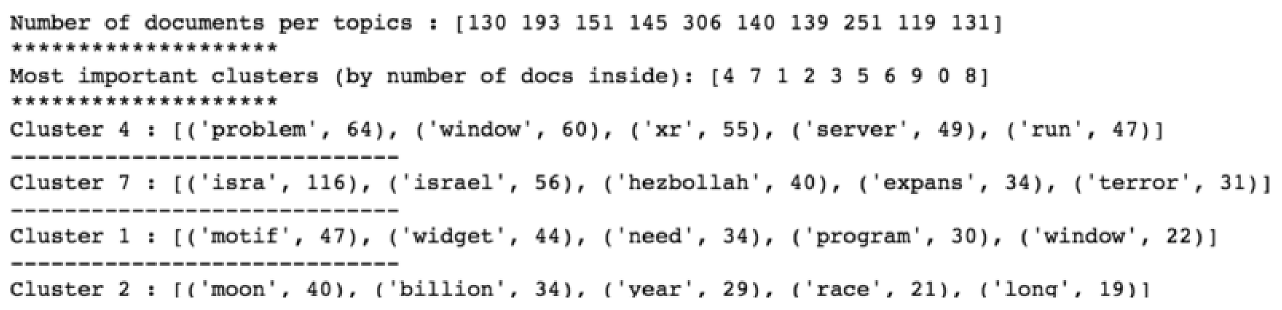
- **alpha** = 0.1 and **beta** = 0.1. Here we kept the default parameters (which work well for several datasets). However, one might want to tune them to improve its topic allocation regarding the completeness and homogeneity of the clusters. Don't hesitate to refer to the original paper 📖 to understand the balance between these two parameters.
- **n_iters** = 30. According to the original paper, GSDMM converges quite fast (about 5 iterations) on several datasets and remain very stable. Hence, 30 iterations is a good default value for any kind of dataset.

Once the model is trained, we want to explore the topics found and check if they are coherent regarding their content 🔎

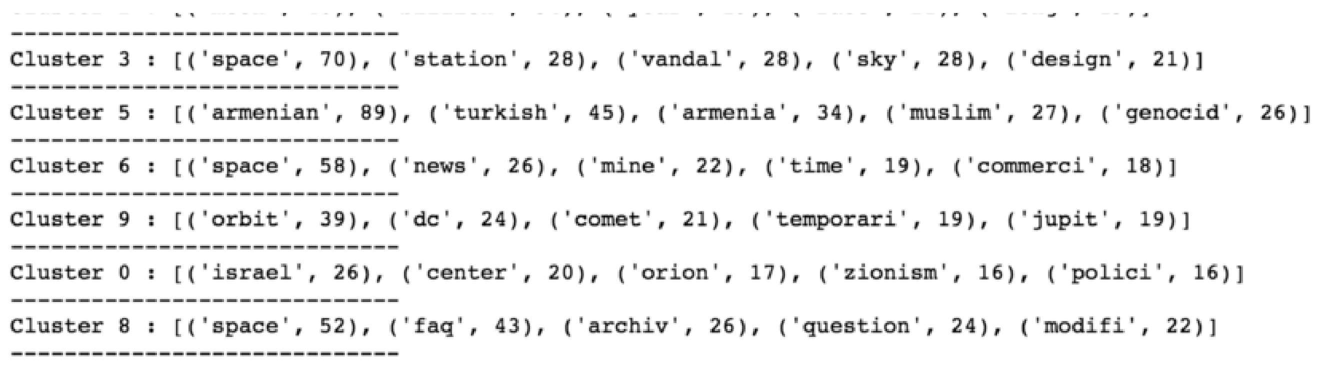
Given that our model has gathered the documents into 10 topics, we must give them a name that will make sense regarding their content. So let's dive into the topics found by our model.

```
doc_count = np.array(mgp.cluster_doc_count)
print('Number of documents per topic :', doc_count)
print('*'*20)

# Topics sorted by the number of document they are allocated to
top_index = doc_count.argsort()[-10:][::-1]
print('Most important clusters (by number of docs inside):',
top_index)
print('*'*20)

# Show the top 5 words in term frequency for each cluster
top_words(mgp.cluster_word_distribution, top_index, 5)
```

The code above display the following statistics that give us insight about what our clusters are made of.



```
Number of documents per topics : [130 193 151 145 306 140 139 251 119 131]
********************
Most important clusters (by number of docs inside): [4 7 1 2 3 5 6 9 0 8]
********************
Cluster 4 : [('problem', 64), ('window', 60), ('xr', 55), ('server', 49), ('run', 47)]
-----------------------------
Cluster 7 : [('isra', 116), ('israel', 56), ('hezbollah', 40), ('expans', 34), ('terror', 31)]
-----------------------------
Cluster 1 : [('motif', 47), ('widget', 44), ('need', 34), ('program', 30), ('window', 22)]
-----------------------------
Cluster 2 : [('moon', 40), ('billion', 34), ('year', 29), ('race', 21), ('long', 19)]
```



```
-----------------------------
Cluster 3 : [('space', 70), ('station', 28), ('vandal', 28), ('sky', 28), ('design', 21)]
-----------------------------
Cluster 5 : [('armenian', 89), ('turkish', 45), ('armenia', 34), ('muslim', 27), ('genocid', 26)]
-----------------------------
Cluster 6 : [('space', 58), ('news', 26), ('mine', 22), ('time', 19), ('commerci', 18)]
-----------------------------
Cluster 9 : [('orbit', 39), ('dc', 24), ('comet', 21), ('temporari', 19), ('jupit', 19)]
-----------------------------
Cluster 0 : [('israel', 26), ('center', 20), ('orion', 17), ('zionism', 16), ('polici', 16)]
-----------------------------
Cluster 8 : [('space', 52), ('faq', 43), ('archiv', 26), ('question', 24), ('modifi', 22)]
-----------------------------
```

Figure 5: 1) Number of documents by cluster (or topic) index. 2) The sorted list of clusters regarding the number of documents they contain. 3) The top 5 words regarding their frequency inside a cluster.

Ideally, the GSDMM algorithm should find the correct number of topics, here 3, not 10. Three explanations come to my mind:

- Find other hyper-parameters to empty smaller cluster (refer to original paper for a deeper understanding of alpha and beta parameters).
- We have both small dataset and vocabulary (about 1700 documents and 2100 words), which may be difficult for the model to extrapolate and distinguish significant difference between topics. As usual, the more data, the better.
- The algorithm might found topics inside the topics. Let me explain. As we well know, one of the topic is about Mideast news. However, the algorithm split this topic into 3 sub-topics: tension between Israel and Hezbollah (cluster 7), tension between Turkish government and Armenia (cluster 5) or Zionism in Israel (cluster 0).

However, even if there are more than 3 found clusters, it's pretty obvious how we can assign them to their respective general topic.

```
# Must be hand made so the topic names match the above clusters
# (Figure 5) regarding their content

topic_dict = {}
topic_names = ['x',
               'mideast',
               'x',
               'space',
               'space',
               'mideast',
               'space',
               'space',
               'mideast',
               'space']
```

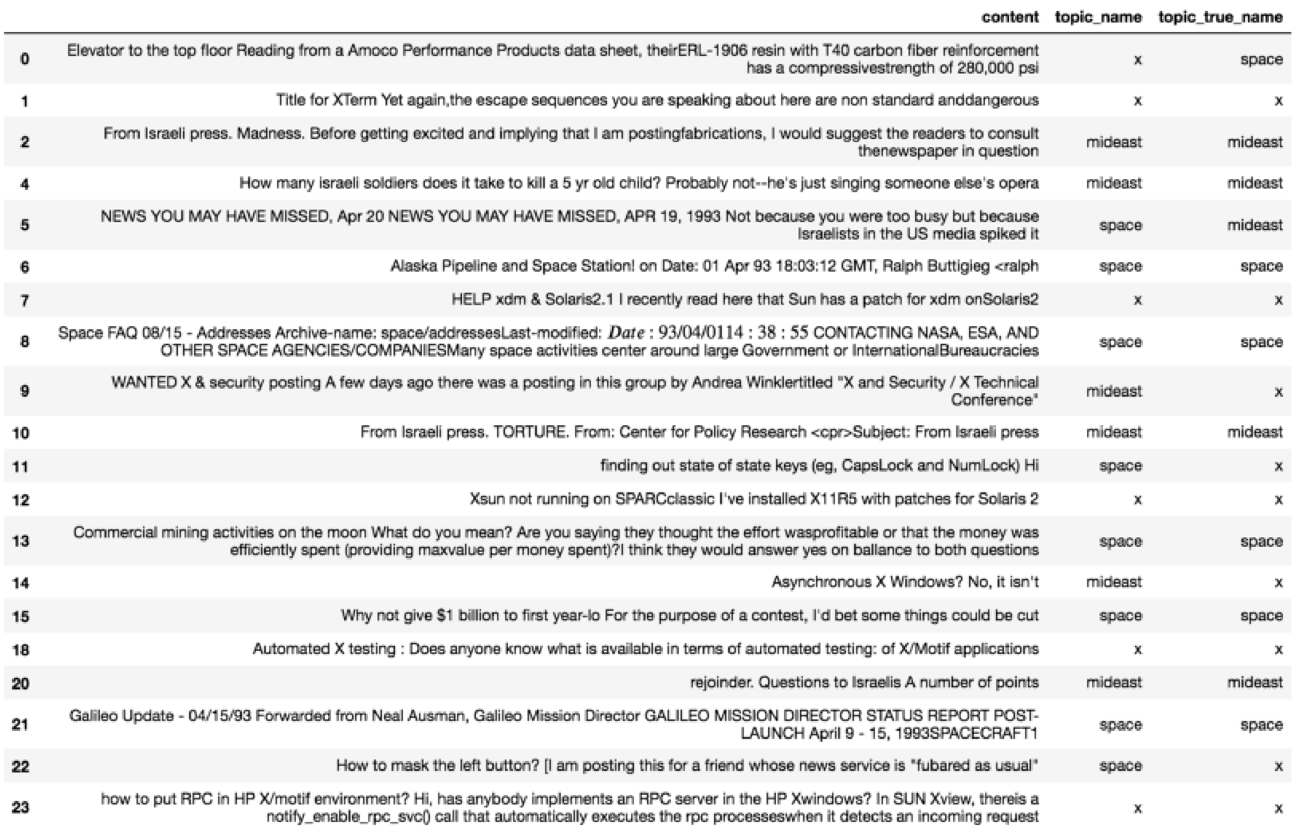
```
for i, topic_num in enumerate(top_index):
    topic_dict[topic_num]=topic_names[i]

df_pred = topic_attribution(tokenized_data, mgp, topic_dict,
threshold=0.4)
```

One might ask what is the *threshold* input parameter of the *topic_attribution* function. Actually, the topics are allocated to a text given a probability and *topic_attribution* is a custom function that allows to choose which threshold (confidence degree) to consider in order to belong to a topic. For example, looking at the highest probability allocation of a topic to a text, if this probability is below 0.4 the text will be allocated in a "Others" topic.

NB: This custom *topic_attribution* function is built upon the original function available in the GSDMM package: *choose_best_label,* which outputs the topic with the highest probability to belong to a document.

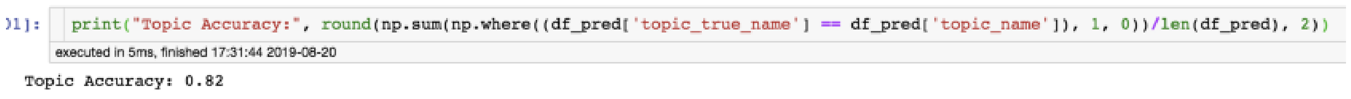
Now it's time to allocate the topic found to the documents and compare them with the ground truth (✅ vs ❌)

```
df_pred[['content', 'topic_name', 'topic_true_name']].head(20)
```



	content	topic_name	topic_true_name
0	Elevator to the top floor Reading from a Amoco Performance Products data sheet, theirERL-1906 resin with T40 carbon fiber reinforcement has a compressivestrength of 280,000 psi	x	space
1	Title for XTerm Yet again,the escape sequences you are speaking about here are non standard anddangerous	x	x
2	From Israeli press. Madness. Before getting excited and implying that I am postingfabrications, I would suggest the readers to consult thenewspaper in question	mideast	mideast
4	How many israeli soldiers does it take to kill a 5 yr old child? Probably not--he's just singing someone else's opera	mideast	mideast
5	NEWS YOU MAY HAVE MISSED, Apr 20 NEWS YOU MAY HAVE MISSED, APR 19, 1993 Not because you were too busy but because Israelists in the US media spiked it	space	mideast
6	Alaska Pipeline and Space Station! on Date: 01 Apr 93 18:03:12 GMT, Ralph Buttigieg <ralph	space	space
7	HELP xdm & Solaris2.1 I recently read here that Sun has a patch for xdm onSolaris2	x	x
8	Space FAQ 08/15 - Addresses Archive-name: space/addressesLast-modified: $Date : 93/04/0114 : 38 : 55$ CONTACTING NASA, ESA, AND OTHER SPACE AGENCIES/COMPANIESMany space activities center around large Government or InternationalBureaucracies	space	space
9	WANTED X & security posting A few days ago there was a posting in this group by Andrea Winklertitled "X and Security / X Technical Conference"	mideast	x
10	From Israeli press. TORTURE. From: Center for Policy Research <cpr>Subject: From Israeli press	mideast	mideast
11	finding out state of state keys (eg, CapsLock and NumLock) Hi	space	x
12	Xsun not running on SPARCclassic I've installed X11R5 with patches for Solaris 2	x	x
13	Commercial mining activities on the moon What do you mean? Are you saying they thought the effort wasprofitable or that the money was efficiently spent (providing maxvalue per money spent)?I think they would answer yes on ballance to both questions	space	space
14	Asynchronous X Windows? No, it isn't	mideast	x
15	Why not give $1 billion to first year-lo For the purpose of a contest, I'd bet some things could be cut	space	space
18	Automated X testing : Does anyone know what is available in terms of automated testing: of X/Motif applications	x	x
20	rejoinder. Questions to Israelis A number of points	mideast	mideast
21	Galileo Update - 04/15/93 Forwarded from Neal Ausman, Galileo Mission Director GALILEO MISSION DIRECTOR STATUS REPORT POST-LAUNCH April 9 - 15, 1993SPACECRAFT1	space	space
22	How to mask the left button? [I am posting this for a friend whose news service is "fubared as usual"	space	x
23	how to put RPC in HP X/motif environment? Hi, has anybody implements an RPC server in the HP Xwindows? In SUN Xview, thereis a notify_enable_rpc_svc() call that automatically executes the rpc processeswhen it detects an incoming request	x	x

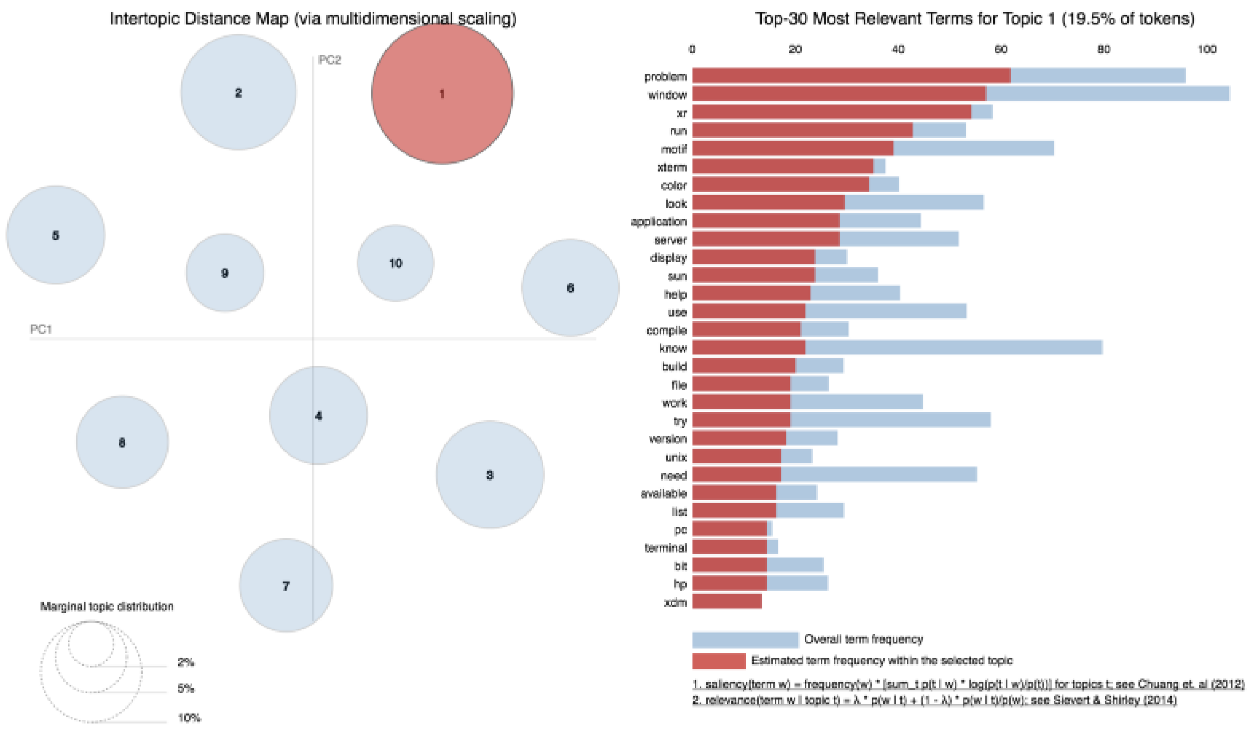
Naively comparing the predicted topics to the true topics we would have had a **82% accuracy**! 🎯
Only with a **9 words average** by document, a small corpus of 1705 documents and very few hyper-parameters tuning!



```
1]:  print("Topic Accuracy:", round(np.sum(np.where((df_pred['topic_true_name'] == df_pred['topic_name']), 1, 0))/len(df_pred), 2))
     executed in 5ms, finished 17:31:44 2019-08-20
  Topic Accuracy: 0.82
```

🙌

It's great to have an efficient model but it is even better if we are able to simply show and interact with its results. To do so, **pyLDAvis** is a very powerful tool for topic modeling visualization, allowing to dynamically display the clusters and their content in a 2-D space dimension.



Screen-shot of pyLDAvis ability. Check out this video to see its full power.

Now it's your turn to try it on your own data (social media comments, online chats' answers…) 💪

Et voilà ! 👌

I would like to thank Rajaa El Hamdani for reviewing and giving me her feedback.

. . .

References and other useful resources
- The original paper of GSDMM
- A nice python package that implements STTM.
- The pyLDAvis library to beautifully visualize topics in a bunch of texts (or any bag-of-words alike data).
- A recent comparative survey of STTM to see other strategies.

PS: For those willing to dive deeper in STTM, there is an interesting further approach (which I have not personally explore for now) called GPU-DMM that has shown SOTA results on Short Text Topic Modeling tasks. In short, GPU-DMM is using pre-trained word embeddings as an external source of knowledge to influence the sampling of words to generate topics and documents.

Machine Learning NLP Data Science Topic Modeling Python