Topic Modeling

https://github.com/josephinefmartell/nlp-topic-modeling-public

Introduction and prior related work

When it comes to text mining, dimensionality can be challenging. The ultimate goal of this project is to use machine learning and automatic natural language processing to create a probabilistic model for abstract themes. During our natural language processing courses, we had the opportunity to explore more in-depth the topic as well as have more information regarding Python, its libraries and Machine Learning.

To succeed in this project, we decided to work on Google Colab. This tool allowed us to work all at the same time in one environment. In addition, we use slack to communicate with each other. In addition, we used internet, and especially shared spaces such as GitHub or Kaggle.

A few weeks ago, we have been given a large CSV file, full of URLs. Once the dataset was cleaned, we decided to scrap the URLs in order to collect the information directly on the website. The challenge was to translate the information into an "attribute-value" array of data that is amenable to algorithmic processing while minimizing the loss of information. In order to build our complex dictionary, a process of tokenization has been used. Then we develop a probabilistic generative model to build the topics from our vocabularies. The model that has been used is Latent Dirichlet Allocation from Gensim library, ideal to model the process of data generation. The last step of our project was to optimize our model to obtain the best results.

In this report, we will explain more in depth the work that has been done with the data. In addition, the elaborate model and some experiments will be detailed. Finally, the results and possible further optimization will be highlighted.

Data

We received a dataset filled with tens of thousand of urls each different from another. As our objective was to create categories based upon those, we identified three necessary steps to follow in order to do so.

First we looked at the urls and their configuration, then started our cleansing by removing all the *http* or *www* as not all the urls had one. This enabled us to standardise the urls and prepare them for the second step which consisted in analysing their titles. In order to do so, we parsed our data by demarcating the various sections of the urls therefore allowing us to classify these various sections. This consisted in our tokenization process. Even though we could have stopped there, we decided to scrap the data from those various urls as we believed it could provide us with insightful information for our classification analysis. Such initiative effectively led to a much more precise classification of the original urls.

Models

In this project, we chose to use a Latent Dirichlet Allocation (LDA). This probabilistic model explains observations using (unobserved) groups of similar data. To go further, it is an unsupervised machine learning algorithm, the output classes are not labeled.

We chose this particular model because it considers that a document is a composition of topics, and that these topics can be distributed on several documents. Thus, the LDA allows overlap topics from one document to another, for a more flexible categorization, and therefore more precise.

We decided to train our model in 4 steps:

- First, we train the LDA only on URLs and see the results.
- Then, on titles (that were previously scrapped from URLs)
- Then on description (that were previously scrapped from URLs)
- In the last section, we train the model on all 3 previous criterias.

Experiments

We decided to run a Latent Dirichlet Allocation model among different kind of techniques to retrieve topics from a text because LDA is a really appropriate technique to maximise the homogenisation of the distribution of topics in a rather small dataset with less variability. Indeed, LSA for instance is way more idiosyncratic by spitting out much more variable topic distribution.

Indeed, LSA assumes a Gaussian distribution of the terms in the documents, which may not be true for all problems. Also, LSA involves SVD, which is computationally intensive

LDA uses lda2vec which is a much more advanced topic modeling which is based on word2vec word embeddings.

Also LDA presents the advantage of giving some freedom to the coder to decide some critical parameters:

Parameters:

- num_topics: *required*. An LDA model requires the user to determine how many topics should be generated. Our document set is small, so we're only asking for three topics.
- id2word: *required*. The LdaModel class requires our previous dictionary to map ids to strings.
- passes: *optional*. The number of laps the model will take through corpus. The greater the number of passes, the more accurate the model will be. A lot of passes can be slow on a very large corpus.

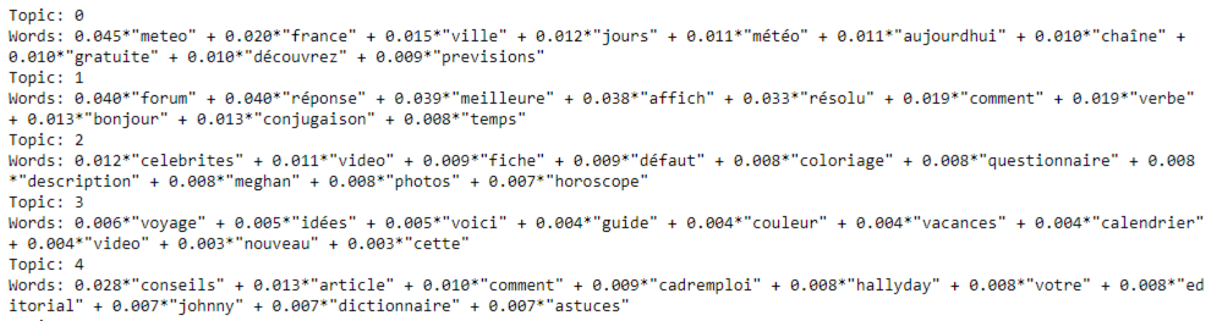
This generates a flat, soft probabilistic clustering of terms into topics and documents into topics.

Results and Conclusion

In order to improve the results, a bunch of improvements were put in place:

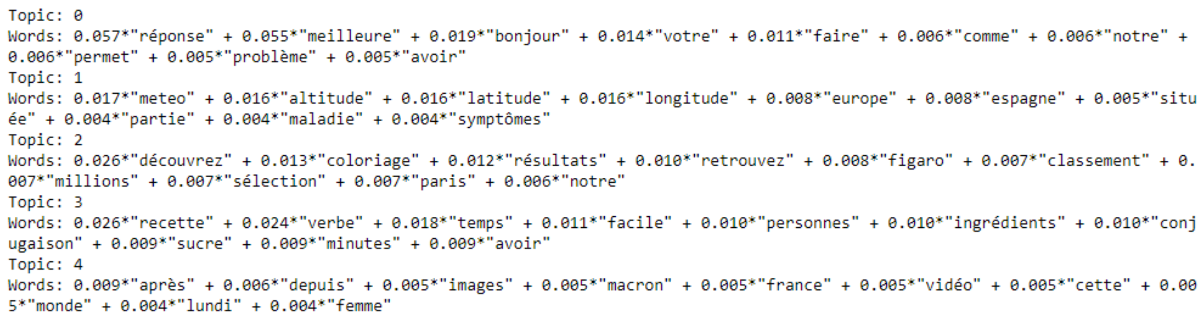
- Defining some unallowed words in order to clean and improve the efficiency of topic identification which plays like a filter.
- The addition of the title improved slightly the result.

Our LDA model is now stored as ldamodel. We can review our topics with the print_topic and print_topics methods:



```
Topic: 0
Words: 0.045*"meteo" + 0.020*"france" + 0.015*"ville" + 0.012*"jours" + 0.011*"météo" + 0.011*"aujourdhui" + 0.010*"chaîne" +
0.010*"gratuite" + 0.010*"découvrez" + 0.009*"previsions"
Topic: 1
Words: 0.040*"forum" + 0.040*"réponse" + 0.039*"meilleure" + 0.038*"affich" + 0.033*"résolu" + 0.019*"comment" + 0.019*"verbe"
+ 0.013*"bonjour" + 0.013*"conjugaison" + 0.008*"temps"
Topic: 2
Words: 0.012*"celebrites" + 0.011*"video" + 0.009*"fiche" + 0.009*"défaut" + 0.008*"coloriage" + 0.008*"questionnaire" + 0.008
*"description" + 0.008*"meghan" + 0.008*"photos" + 0.007*"horoscope"
Topic: 3
Words: 0.006*"voyage" + 0.005*"idées" + 0.005*"voici" + 0.004*"guide" + 0.004*"couleur" + 0.004*"vacances" + 0.004*"calendrier"
+ 0.004*"video" + 0.003*"nouveau" + 0.003*"cette"
Topic: 4
Words: 0.028*"conseils" + 0.013*"article" + 0.010*"comment" + 0.009*"cadremploi" + 0.008*"hallyday" + 0.008*"votre" + 0.008*"ed
itorial" + 0.007*"johnny" + 0.007*"dictionnaire" + 0.007*"astuces"
```

What does this mean? Each generated words is separated by a comma. Within each set of topics are the most probable words to appear in that topic. Even though our document set is not too voluminous the model is reasonable. Some things to think about: - metéo, france, ville and so on make sense together whereas before the results were less meaningful:



```
Topic: 0
Words: 0.057*"réponse" + 0.055*"meilleure" + 0.019*"bonjour" + 0.014*"votre" + 0.011*"faire" + 0.006*"comme" + 0.006*"notre" +
0.006*"permet" + 0.005*"problème" + 0.005*"avoir"
Topic: 1
Words: 0.017*"meteo" + 0.016*"altitude" + 0.016*"latitude" + 0.016*"longitude" + 0.008*"europe" + 0.008*"espagne" + 0.005*"situ
ée" + 0.004*"partie" + 0.004*"maladie" + 0.004*"symptômes"
Topic: 2
Words: 0.026*"découvrez" + 0.013*"coloriage" + 0.012*"résultats" + 0.010*"retrouvez" + 0.008*"figaro" + 0.007*"classement" + 0.
007*"millions" + 0.007*"sélection" + 0.007*"paris" + 0.006*"notre"
Topic: 3
Words: 0.026*"recette" + 0.024*"verbe" + 0.018*"temps" + 0.011*"facile" + 0.010*"personnes" + 0.010*"ingrédients" + 0.010*"conj
ugaison" + 0.009*"sucre" + 0.009*"minutes" + 0.009*"avoir"
Topic: 4
Words: 0.009*"après" + 0.006*"depuis" + 0.005*"images" + 0.005*"macron" + 0.005*"france" + 0.005*"vidéo" + 0.005*"cette" + 0.00
5*"monde" + 0.004*"lundi" + 0.004*"femme"
```

Clearly, one can see that adding the titles helped enriching the data for the algorithm to classify better the article by seizing the general context. Before the results were mainly related to the distribution of those word within the article which makes sense but doesn't provide sufficient insights on the real topic of the article.