Topic Modeling with LDA model

In this project we have a dataset of 5K articles. The purpose is to prepare that dataset and train LDA model for topic modeling. Finally, predict 3 topics for any given article, those 3 topics are the ones with highest probability to be well fitted on the article.
After process the data and train the model, the results show that the optimal number of topics for those articles dataset is 12 topics.

1. The project consist of 2 models (baseline model "lda_all_data_model" and best model "bettter_lda_model") and 7 notebooks written in python, those scripts are:
Dataset → contains a class Dataset, it is used to read and set the data in the required format to be ready for the next step of processing

Data_cleaning → contain Data_Cleaning class, it has all the methods required to clean the data. The cleaning process is as the following:

1. Convert text to lower case
2. remove any newline or tabs
3. remove any email format, to be sure that it will not affect on produced topics keywords
4. remove punctuations

Data_preprocessing → contain Data_Preprocessing class, it has methods which written for applying preprocessing steps which are:

1. apply tokenization
2. remove English stop words
3. apply lemmatization
4. create a dictionary to give every unique word a unique id
5. create a corpus to count he number of rebeating each unique word through out the data (b y apply BAG OF WORDS "BoW")
6. Also, I write a method for stemming. But I didn't include it in the preprocessing step, as stemming returns words to their initial format and that can badly affect the topics quality as the models will find many word in the same shape while they in original text have different meanings.
    - ⇨ I build a dictionary which contains all unique words and their ids, also build a corpus which contains the words ids and their Bow, because the model require the data in those two formats to train on. And we can get the model topics by using the dataset corpus.

Utils → That notebook contains all the helper methods which are required to successfully finish the models training and also draw the graphs to help in better

model performance analysis. Like see the most common words in the dataset, see the distribution of the model generated topics, train the model, calculate the model performance, save model,….

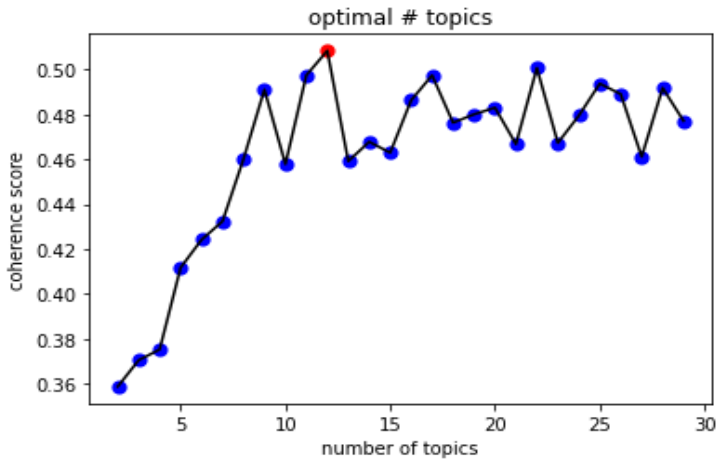
Test_tuils → notebook contains all the helper functions required to test the model (model prediction). Like make the model provide top three topics that fit on the given article, and print those topics and their probabilities (The model takes the topic after preprocesses it assign topics to article with different probabilities).

Topic_modeling → in that notebook, I use all the previous three classes to read, clean and prepare the data. Then I used the utils functions to train the model, visualize its topics distribution, also measure its performance. First train a simple LDA baseline model, then train many other LDA model with hyperparameters tuning to model which has performance better than the baseline one.

Test_lda_model → through that notebook, load the saved model (provide path of any model (baseline or best one as you want) to test its performance on unseen articles. All you have to do is pass the text of any article.

Metrics used to measure the performance: I write 2 functions to measure lda model performance in the utils notebook. Those two metrics are (perplexity and cv coherence score). But I use only coherence score. And that because it can measure how interpretable the topics are to humans. As we know that lda model return topics in the form of N words with the highest probability of belonging to that particular topic. And the coherence score measures how similar these words are to each other. And the cv uses words to create vectors using their co-occurrences, then it calculate the score using cosine similarity and NPMI.

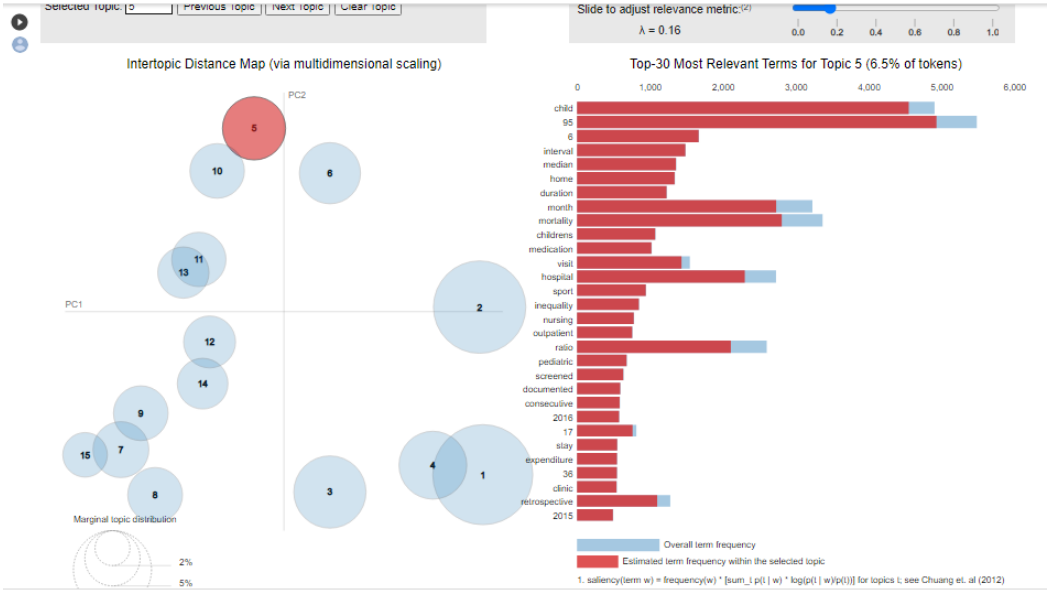
Process: First, train a simple lda model with #topics = 15. Then go through a loop to train number of models (28) with different number of topics (from 2 to 30) for each model, and setting the rest of parameter to the following values: Random state = 24, chunk size = 500, pass = 15. Then calculate the coherence score for each model of them, and from the model with best score we take its #topics as the optimal one. From the following graph, the optimal #topics are **12**.



#topics for every model

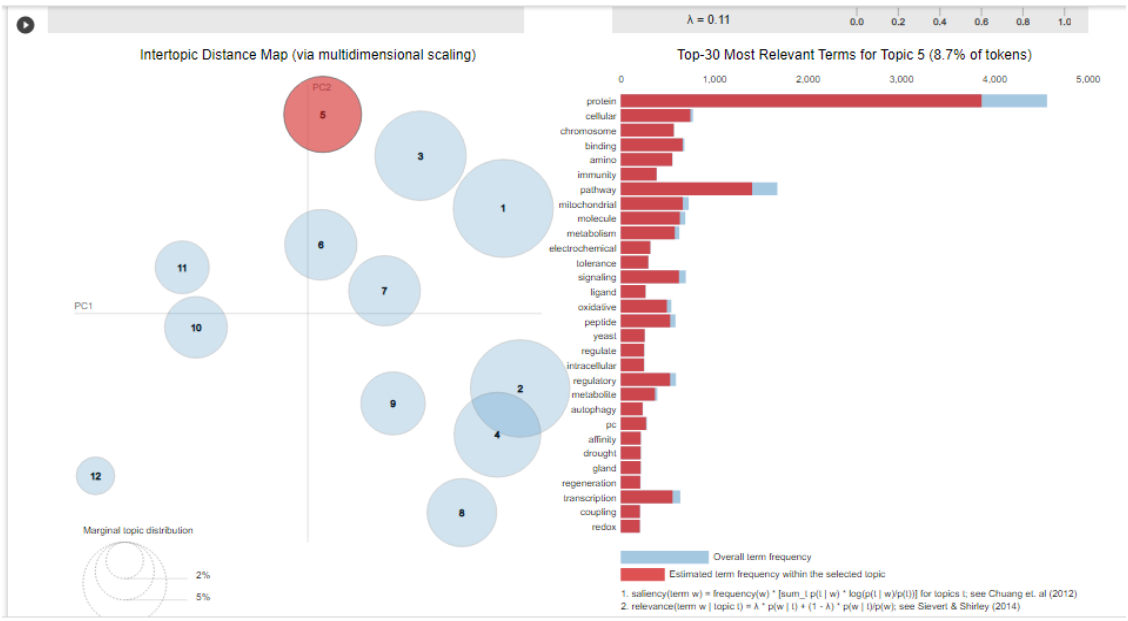
Evaluation and how to improve the data and performance:

The baseline model has coherence score = 0.43, and there is many intersections among generated topics like topics(11,13), topics(15,7,9) and more. That is not good as we need a model which can provide separate topics without overlapping among them.



Topics distribution from baseline model

Then, after applying hyperparameter tuning to get a better model, we reach a coherence score = 0.50. So, we have a model with enhancement in its performance with +0.7. And there is good enhancement on the generated topics, as there is much less number of intersections among topics.



Topics distribution from baseline model

Assign 3 topics for each input document

```
# test any other text
text = """In recent years, researchTrusted Source has suggested that levels of mental stress
influence gut bacteria composition and play a key role in IBS via the gut-brain axis.
"""
test(text, lda_model, all_topics)
```

```
topic nuber:  1  with index:  12  and probability:  0.15219547  is:  model method data using system result used analysis based performance
topic nuber:  2  with index:  6  and probability:  0.12665182  is:  health study research review intervention data quality patient service social
topic nuber:  3  with index:  5  and probability:  0.11194792  is:  study risk year among health associated age child factor woman
```

Top 3 topics assigned o unseen article.

To improve the data and model performance more, we can apply the following:

- Remove emoji if exists, as it might affect on the topics quality
- Apply n_grams and IFIDF on dataset instead of only BoW.Try
- Train a model many times with different values for the above parameters and alpha.
- Use LDA Mallet model. Mallet version of lda provide topics with better quality.

Used tools:

- Google colab, Google drive, python.
- Gensim for the model, CoherenceModel for coherence score
- Wordcloud, pyLDAvis, matplotlib for graphs.
- Pikle to save and load the model.

⇨ You will find comments on code and more discussion on the data and models performance.