Topic Modeling with LDA and Spark

Naimisha Churi
801257788

What is Topic Modeling?

Topic Modeling is a form of statistical modeling used to identify the abstract "themes" that appear in a group of texts – a way to obtain recurring patterns of words in textual material.

It is a popular text-mining technique for locating latent semantic patterns in a text body. The basic intuition behind this is that certain words would occur in a document more or less frequently if it were about a specific topic.

Motivation

- More than 80% of the data generated in the world is unstructured or semi-structured; most of it is text data. The motivation to select this topic was to work with text data and gain more experience with it.
- Topic modeling is one of the topics that caught my attention as it has many applications and can be used to save time while obtaining information as it extracts what the given text is about.
- Spark is a technology that made processing big data faster and more efficient. It is widely used in the industry to process large amounts of data
- I see this project as a learning experience and an opportunity to show and polish my skills while gaining experience in implementing a machine-learning algorithm

Applications and Benefits

Topic models allow us to answer big-picture questions quickly, cheaply, and without human intervention. Once trained, they provide a framework for humans to understand document collections both directly by “reading” models or indirectly by using topics as input variables for further analysis. Training a Topic model on a huge corpus of documents will help understand the essence and the content of the corpus efficiently and at a relatively low cost. Following are a few applications of Topic Modeling

- Discovering hidden topical patterns that are present across the collection
- Annotating documents according to these topics
- Using these annotations to organize, search and summarize texts
- The output obtained from the model can be used to perform further analysis of the data and for applications like Text Summarization and Sentiment analysis

LDA - Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is an unsupervised clustering technique that is commonly used for text analysis. It's a type of Topic Modeling algorithm which works on

the intuition that the documents consist of topics and those topics are a collection of words.
LDA is a generative probabilistic model for collections of discrete data. It is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document.
Working:

- Goal - Learn the topic mix in every document and the word mix in each topic
- Manually choose the number of topics 'n' as the number of clusters
- LDA then reads the entire corpus
- Randomly assigns each word in each document to one of the n topics
- Go through every word and its topic allocation in each document and look at -
    - How often this topic occurs in the document
    - How often the word appears in the topic overall
    - Based on this information assign the word a new topic
    - Repeat this process iteratively

Distributed LDA:
In this version of LDA, the data and parameters are distributed over distinct processors. This approach is adopted from the Distributed Inference for Latent Dirichlet Allocation paper. The Approximate Distributed LDA model (AD-LDA), is a simple implementation of LDA on each processor, and simultaneous Gibbs sampling is performed independently on each of the P processors as if each processor thinks it is the only one. At the start of each iteration, all of the processors have the same set of counts. However, as each processor starts sampling, the global count matrix is changing in a way that is unknown to each processor.
Input - bag of words document representation, vocabulary, number of topics, number of topic words, number of iterations.
Output - List of topic words for the specified number of topics.

Data -

- About 210k news headlines from HuffPost from 2012 to 2022 are included in this collection.
- It is used as a benchmark for a number of computational linguistic problems and is one of the largest news databases.
- As of 2018, HuffPost ceased maintaining a sizable archive of news items, making it impossible to gather such a dataset today.
- The data is about 87.3MB in size
- This data has 6 features out of which I have used 'category' for balancing the dataset and 'headline' as the set of documents.

Tasks Involved and Approach

Following are the steps and experiments I did for the project -

- Obtaining the data
  Initially, I decided to work with the CNBC News Dataset from data.world. After some research, it came to my attention that for Topic modeling to produce good results, it needs a higher number of input data points than was present in the dataset. Eventually, after running a few experiments with the sklearn's LatentDirichletAllocation library, I landed on the News Category Dataset on Kaggle which contains titles and categories of over 200,000 news articles in JSON format.
- Performing EDA
  The next step was to perform exploratory data analysis and get to know the data. The data had 6 columns and 209527 entries of News articles that fell under 42 categories ranging from 'WORLD NEWS' to 'HEALTHY LIVING' and 'ARTS'. I selected the 'TECH', 'SPORTS', 'HEALTHY LIVING', 'STYLE', and 'ENVIRONMENT' which seem to have the least amount of logical overlap to fix the number of Topics.
- Cleaning and preprocessing the data
  The data obtained after discarding the other categories was unbalanced and had words that did not contribute to the content. Spark has a variety of operations you can perform on the data for preprocessing and exploration, and it was fun to learn them.
- Implementing Latent Dirichlet allocation algorithm
  To run LDA on the data, first, it needs to be converted into a vector format. TF-IDF seemed to be the obvious choice and since it accounts for the relevance of words in a corpus, there was no need to remove the stop-words from the data. To perform this operation, I used sklearn's CountVectorizer a pre-existing library. It converts the documents to a sparse matrix of token counts with the number of features will equal to the vocabulary size.
  For the LDA algorithm, first, the parameters and matrices are initialized. Then for the specified number of iterations, for every word in every document, we decrement the counts for the word with the associated topic, sample a new topic from a multinomial according to the formula, set a new topic, and increment counts for that topic.
- Train the model
  Next, I trained the model to obtain the topic words for various combinations of parameters. The implemented LDA function takes in 5 parameters
    - Collection of documents
    - Vocabulary
    - Number of Topics

    - Number of Topic Words
    - Number of iterations
- Observing and analyzing the results obtained

Results

Following are the results obtained with the LDA model by tweaking its parameters which are as follows:

- Collection of documents
- Vocabulary
- Number of Topics
- Number of Topic Words
- Number of iterations

```
1 LDA(docs, vocab, 3, 10, 10)
```

```
Topic #0: week photos fashion best apple day style beauty people animal
Topic #1: apple just health climate iphone people change week trump says
Topic #2: olympic olympics world game team facebook google says just twitter


Time taken to run LDA for 3 topics and 10 topic words for 10 iterations: 13.559024095535278ms
```

```
1 LDA(docs, vocab, 5, 10, 10)
```

```
Topic #0: best 10 things facebook beauty fashion week people years 2016
Topic #1: week apple iphone photos climate animal just fashion list change
Topic #2: health just says climate olympic world future study sports game
Topic #3: people facebook health heres ways like help just google world
Topic #4: olympics trump olympic nfl team gold zika wins says rio


Time taken to run LDA for 5 topics and 10 topic words for 10 iterations: 13.42873215675354ms
```

- We can observe that the set of documents has information about the Rio Olympics, sports, climate change, tech companies like Facebook, Apple, Fashion, Politics, etc, and also some articles about healthy living
- LDA requires less number of iterations to converge. At 10 iterations, we can see that the topics give us an idea about the content of the data.
- The topics generated by the first model are more distinct and have a lesser intersection compared to the 5-topic count because there are fewer topics.
- The model with 5 topics tells more about the data than the 3 topics even though there is a smaller overlap in the topics generated by the first model.

```
1 LDA(docs, vocab, 5, 5, 10)
```

```
Topic #0: week apple photos iphone animal
Topic #1: health climate change care cancer
Topic #2: olympic best just beauty game
Topic #3: like just people heres facebook
Topic #4: team nfl trump player says


Time taken to run LDA for 5 topics and 5 topic words for 10 iterations: 13.126483917236328ms
```

```
1 LDA(docs, vocab, 10, 5, 10)
```

```
Topic #0: things people care health cancer
Topic #1: week photos best red animal
Topic #2: climate health change heres like
Topic #3: just need help ways jenner
Topic #4: just facebook time game health
Topic #5: olympics olympic winter rio says
Topic #6: trump nfl super says climate
Topic #7: apple iphone week olympic rumors
Topic #8: world abuse facebook million york
Topic #9: say use google crisis years


Time taken to run LDA for 10 topics and 5 topic words for 10 iterations: 13.427200078964233ms
```

- The model with 10 topics produces topics that have a lot of overlap and common words and does not have a lot of extra information about the corpus than the models with fewer topics

```
1 LDA(docs, vocab, 3, 10, 5)
```

```
Topic #0: week like photos 10 people things fashion animal health best
Topic #1: apple iphone just best week google climate time says day
Topic #2: olympic facebook health olympics game says world just team climate


Time taken to run LDA for 3 topics and 10 topic words for 5 iterations: 6.69437313079834ms
```

```
1 LDA(docs, vocab, 5, 10, 5)
```

```
Topic #0: week photos apple olympic animal says world iphone nfl baby
Topic #1: week apple best just climate iphone health change 10 cancer
Topic #2: world day climate like health need google fashion internet olympics
Topic #3: heres facebook just like way life apple google tips things
Topic #4: olympics people fashion health team time rio twitter says just


Time taken to run LDA for 5 topics and 10 topic words for 5 iterations: 6.722424030303955ms
```

```
1 LDA(docs, vocab, 3, 10, 50)
```

```
Topic #0: week best fashion photos day like make just look beauty
Topic #1: health olympic olympics team gold care people cancer just rio
Topic #2: climate facebook google apple change trump nfl game twitter says


Time taken to run LDA for 3 topics and 10 topic words for 50 iterations: 66.53231382369995ms
```

```
1 LDA(docs, vocab, 5, 10, 50)
```

```
Topic #0: best fashion week beauty style looks look dress red like
Topic #1: apple climate google iphone change facebook week photos million study
Topic #2: trump nfl says health trumps twitter michael donald report abuse
Topic #3: people health dont make cancer life things need know women
Topic #4: olympic olympics game gold world team win nba super watch


Time taken to run LDA for 5 topics and 10 topic words for 50 iterations: 67.08164310455322ms
```

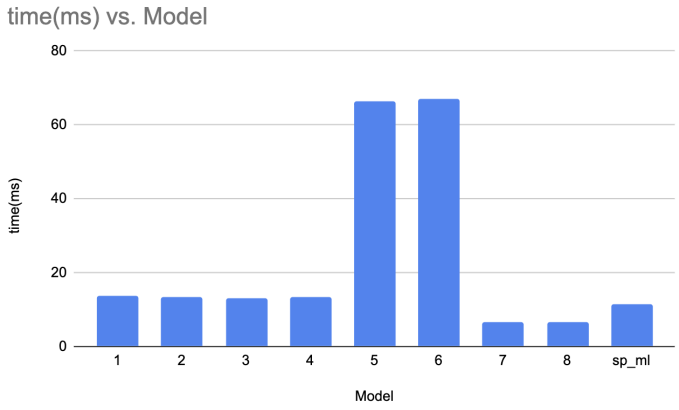
- We can see that the topics are better formed and have less overlap and have converged better after 50 iterations and that they have not and don't make a lot of sense just after 5 iterations.
- Running the code for 50 iterations requires a significant amount of more time.

Topics were obtained after 10 iterations of the pre-existing SparkMLlib implementation of LDA.

```
Topic #0: time make like trump heres look 5 twitter black million
Topic #1: could health world people may care iphone need cancer get
Topic #2: us olympic first olympics team gold game winter super rio
Topic #3: week  photos fashion animal nfl day baby york way
Topic #4: climate facebook change apple says beauty dont help best google

Time taken to run LDA for 5 topics and 10 topic words for 50 iterations: 11.521415948867798ms
```

Chart displaying the time taken by all the models to run respectively.
The last column is the time required by the MLlib's LDA algorithm implementation which ran for 50 iterations, thus displaying the scope of improvement of my model.



Word cloud of all the words in the corpus and as we can see few of the frequent words are in different topic words

Challenges faced -

- SparkNLP - it is licensed and the free version cannot be installed on the EMR cluster
- The implementation of LDA with Gibbs sampling - took a lot of time and effort and needs a lot of refinement but was fun to do.

External packages used -

- Sci-kit Learn's Count Vectorizer - for TF-IDF and vector representation of the documents

Aspects of the project achieved -

- Implementation of LDA algorithm
- EDA and data processing using Spark
- Analysis and comparison of the LDA model with a pre-existing model built in the spark MLlib

Future Scope -

- Using SparkNLP to perform data preprocessing
- Improving the performance of the model
- Deploy the model in the cloud

References -

- https://en.wikipedia.org/wiki/Topic_model
- https://mimno.infosci.cornell.edu/papers/2017_fntir_tm_applications.pdf
- https://www.kdnuggets.com/2016/07/text-mining-101-topic-modeling.html
- https://datascienceplus.com/topic-modeling-and-latent-dirichlet-allocation-lda/

- https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf
- https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.clustering.LDA.html