TOPIC MODELLING

Sanchayan Bhunia

PROBLEM

Data : https://www.Kaggle.com/

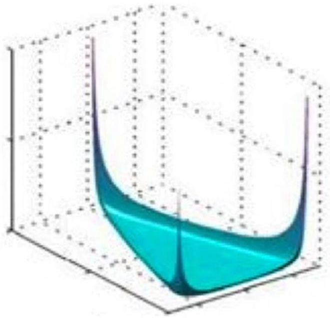
- There are two csv files – NEWS_Content_100, NEWS_Content_4550
- First one is for test just 100 news articles . Second one is the complete 4500 news articles.
- No of cloumn : Index , Content
- Data analysis : TF-IDF , LDA, Analysis on RDD

TF-IDF

- TF stands for term frequency.
- Frequency of a word in a given article.
- IDF stands for inverse document frequency.
- Frequency of the same word in the entire corpus.
- TF-IDF score is defined as

$$Tf - IDF = \log\left(\frac{\{Term\ frequency\}}{\{Document\ Frequency\}}\right)$$

LDA

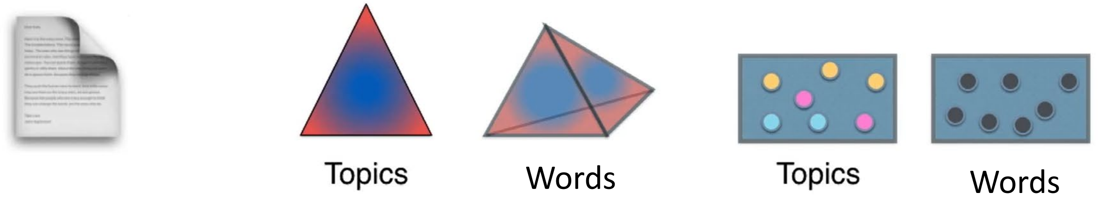


Dirichlet Distribution

- LDA stands for Latent Dirichlet Allocation
- Basically a machine that makes documents
- The parameters are the tuners. Tune them to get documents similar to our real documents.
- First two Dirichlet pdf. Rest are Multinomial distributions.

Probability of a document

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, P(W_{j,t} \mid \varphi_{Z_{j,t}})$$



DATA ANALYSIS AND OPERATIONS

- Use of RDD and TF-IDF and LDA methods
- Operations :-
- Map function
- Transform function
- Filter function
- Join
- Split
- Words function
- zipWithIndex function
- Countvectorizer function

METHDOLOGY

- Operations :

- First we use to clean/pre processing (strip(),split()) on the data .
- We use of two main methods – TF-IDF(Term Frequency and Inverse Document Frequency) for finding the important and unique word in the article.
- LDA(latent Dirichlet Allocation) for finding out the what is the most important words in article ? Based on probability.

METHDOLOGY

➢ Libraries:

- Pandas
- Sparkcontext from pyspark
- SQLContext from pyspark.sql
- Stopwords from nltk.corpus
- CountVectorizer , IDF from pyspark.ml.feature
- Vector , Vectors from pyspark.mllib.linalg
- LDA , LDAModel from pyspark.ml.clustering

Pre Processing

- **Pre processing** :

```
contents = data.rdd.map(lambda x : x['Content']).filter(lambda x: x is not None)
StopWords = stopwords.words("english")
tokens = contents
    .map( lambda document: document.strip().lower())
    .map( lambda document: re.split(" ", document))
    .map( lambda word: [x for x in word if x.isalpha()])
    .map( lambda word: [x for x in word if len(x) > 3] )
    .map( lambda word: [x for x in word if x not in StopWords])
    .zipWithIndex()
df_txts = sqlContext.createDataFrame(tokens, ["list_of_words",'index'])
```

RESULTS

- **Final result :**

datadf = data.selectExpr("_c0 as Index", "Content as Content")

datadf.show()

result = datadf.join(transformed,on="index",how="left")

result.show()

```
-----+--------------------+--------------------+--------------------+--------------------+
 Index|             Content|       list_of_words|            features|   topicDistribution|
-----+--------------------+--------------------+--------------------+--------------------+
    26|Remain camp will ...|[remain, camp, re...|(4000,[0,1,2,3,4,...|[2.02047473036179...|
    29|Noel Gallagher: W...|[noel, never, mad...|(4000,[1,2,3,5,10...|[0.44357768228427...|
    65|Drill, baby, dril...|[tillerson, state...|(4000,[0,1,5,6,12...|[0.06905523892431...|
    19|European Union re...|[european, union,...|(4000,[0,1,2,3,4,...|[8.47653578986475...|
    54|Brad and Angelina...|[brad, angelina, ...|(4000,[0,3,5,8,13...|[1.68914992076396...|
-----+--------------------+--------------------+--------------------+--------------------+
only showing top 5 rows

user12@master:~/topicModelling$
```

SOURCE CODE (Interesting part)

- There are two methods we use in this for developing TF-IDF and LDA to describe a word frequency in each sentence , whole news article and words in terms of vectors

Code:

```
# TF
cv = CountVectorizer(inputCol="list_of_words", outputCol="raw_features", vocabSize=5000, minDF=10.0)
cvmodel = cv.fit(df_txts)
result_cv = cvmodel.transform(df_txts)
# IDF
idf = IDF(inputCol="raw_features", outputCol="features")
idfModel = idf.fit(result_cv)
result_tfidf = idfModel.transform(result_cv)
#result_tfidf.show()
df_model=result_tfidf.select('index','list_of_words','features')
#df_model.show()
```

Result of TF-IDF :

user12@master: ~/topicModelling

```
user12@master:~/topicModelling$ python3 code.py
20/06/12 05:28:33 WARN NativeCodeLoader: Unable to load native-
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For Spark
+-----+--------------------+--------------------+
|index|       list_of_words|            features|
+-----+--------------------+--------------------+
|    0|[defiance, fines,...|(2,[0,1],[0.32466...|
|    1|[hillary, lead, p...|           (2,[],[])|
|    2|[greatest, ally, ...|           (2,[],[])|
|    3|[fight, cruz, rub...|(2,[0,1],[0.32466...|
|    4|[voting, america,...|(2,[0,1],[0.64932...|
+-----+--------------------+--------------------+
only showing top 5 rows
```

SOURCE CODE (Intersting part)

- **Code** : creating LDA model

```
num_topics = 5
max_iterations = 100
lda_model = LDA(k=num_topics, maxIter=max_iterations)
model=lda_model.fit(df_model)
model.describeTopics(5).show()
model.describeTopics().first()
transformed = model.transform(df_model)
transformed.show()
```

Result of LDA :

```
+-----+--------------------+--------------------+
|topic|         termIndices|         termWeights|
+-----+--------------------+--------------------+
|    0|[130, 424, 91, 13...|[0.01375540436115...|
|    1|[29, 11, 68, 197, 9]|[0.01446529116982...|
|    2|[6, 202, 61, 40, 29]|[0.01409162314795...|
|    3|[22, 9, 20, 80, 200]|[0.00442048187882...|
|    4|[632, 121, 332, 3...|[0.00537052953235...|
+-----+--------------------+--------------------+
```

```
20/06/12 05:34:57 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.Nat
20/06/12 05:34:57 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.Nat
+-----+--------------------+--------------------+--------------------+--------------------+
|Index|             Content|       list_of_words|            features|   topicDistribution|
+-----+--------------------+--------------------+--------------------+--------------------+
|   26|Remain camp will ...|[remain, camp, re...|(2,[0,1],[10.0644...|[0.85947526534940...|
|   29|Noel Gallagher: W...|[noel, never, mad...|(2,[1],[0.7439195...|[0.63744454434514...|
|   65|Drill, baby, dril...|[tillerson, state...|(2,[0,1],[0.32466...|[0.79597667203736...|
|   19|European Union re...|[european, union,...|(2,[0,1],[11.6877...|[0.86078031487530...|
|   54|Brad and Angelina...|[brad, angelina, ...|(2,[0],[0.6493221...|[0.31001353684829...|
+-----+--------------------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Performance

Time in seconds	Local Machine	DLTM Cluster	Performance	Gain/Loss
cleaning time	0.9333427	1.5145878	0.5812451	↓
TF-IDF time	2.3047293	3.71139456	1.40666526	↓
LDA time	144.979599	65.388761	79.590838	↑
total time	152.3190864	72.07446	80.2446264	↑

Local Machine Configuration

Device specifications

Device name	Odin
Processor	Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz (4 physical cores 8 logical cores after hyperthreading)
Installed RAM	16.0 GB (15.8 GB usable)
Device ID	EB14F9C1-FE25-4FCC-9981-596DB9CB1A7D
Product ID	00330-80127-04068-AA239
System type	64-bit operating system, x64-based processor
Pen and touch	Touch support with 10 touch points

Rename this PC

Windows specifications

Edition	Windows 10 Pro
Version	2004
Installed on	10 June 2020
OS build	19041.329
Experience	Windows Feature Experience Pack 120.2202.130.0

DLTM Cluster Configuration

Architecture:	x86_64
CPU op-mode(s):	32-bit, 64-bit
Byte Order:	Little Endian
CPU(s):	8
On-line CPU(s) list:	0-7
Thread(s) per core:	1
Core(s) per socket:	8
Socket(s):	1
NUMA node(s):	1
Vendor ID:	GenuineIntel
CPU family:	6
Model:	63
Intel(R) Xeon(R) CPU E5-4620 v3 @ 2.00GHz	Intel(R) Xeon(R) CPU E5-4620 v3 @ 2.00GHz

References

- Grokking Machine Learning, Luis G. Serrano. ISBN 9781617295911
- Demonstration by Luis G. Serrano. YouTube Link
- Topic modeling using Latent Dirichlet Allocation(LDA) and Gibbs Sampling explained! Medium Link
- Latent Dirichlet Allocation (LDA) for Topic Modelling Presentation by Sina Miran, University of Maryland. Link
- Topic Modelling: an explanation. Towards data science link

Thank you