Topic Modelling Using Latent Dirichlet Allocation

About Latent Dirichlet Allocation :
LDA assumes documents are produced from a mixture of topics. Those topics then generate words based on their probability distribution. Given a dataset of documents, LDA backtracks and tries to figure out what topics would create those documents in the first place.

LDA is a matrix factorization technique. In vector space, any corpus (collection of documents) can be represented as a **document-term matrix**. The following matrix shows a corpus of N documents D1, D2, D3 … Dn and vocabulary size of M words W1,W2 .. Wn. The value of i,j cell gives the frequency count of word Wj in Document Di.

	W1	W2	W3	Wn
D1	0	2	1	3
D2	1	4	0	0
D3	0	2	3	1
Dn	1	1	3	0

LDA converts this Document-Term Matrix into two lower dimensional matrices – M1 and M2. M1 is a **document-topics matrix** and M2 is a **topic – terms matrix** with dimensions (N, K) and (K, M) respectively, where N is the number of documents, K is the number of topics and M is the vocabulary size.

	K1	K2	K3	K
D1	1	0	0	1
D2	1	1	0	0
D3	1	0	0	1
Dn	1	0	1	0

	W1	W2	W3	Wm
K1	0	1	1	1
K2	1	1	1	0
K3	1	0	0	1
K	1	1	0	0

Notice that these two matrices already provides topic word and document topic distributions, However, these distribution needs to be improved, which is the main aim of LDA. LDA makes use of **sampling techniques** in order to improve these matrices.

It Iterates through each word “w” for each document “d” and tries to adjust the current topic – word assignment with a new assignment. A new topic “k” is assigned to word “w” with a probability P which is a product of two probabilities p1 and p2.

For every topic, two probabilities p1 and p2 are calculated. P1 – p(topic t / document d) = the proportion of words in document d that are currently assigned to topic t. P2 – p(word w / topic t) = the proportion of assignments to topic t over all documents that come from this word w.

The current topic – word assignment is updated with a new topic with the probability, product of p1 and p2 . In this step, the model assumes that all the existing word – topic assignments except the current word are correct. This is essentially the probability that topic t generated word w, so it makes sense to adjust the current word’s topic with new probability.

After a number of iterations, a steady state is achieved where the document topic and topic term distributions are fairly good. This is the convergence point of LDA.

Parameters of LDA

Alpha and Beta Hyperparameters – alpha represents document-topic density and Beta represents topic-word density. Higher the value of alpha, documents are composed of more topics and lower the value of alpha, documents contain fewer topics. On the other hand, higher the beta, topics are composed of a large number of words in the corpus, and with the lower value of beta, they are composed of few words.

Number of Topics – Number of topics to be extracted from the corpus.

Number of Topic Terms – Number of terms composed in a single topic. It is generally decided according to the requirement. If the problem statement talks about extracting themes or concepts, it is recommended to choose a higher number, if problem statement talks about extracting features or terms, a low number is recommended.

Number of Iterations / passes – Maximum number of iterations allowed to LDA algorithm for convergence.

Implementation of the algorithm:

The user shall enter the path of the PDF file and he can deduce the topic by studying the probability distribution of the terms in the topic. He will get the probability distribution of the top four words of the topic which can further be changed by changing the value of the variable num_words.

```
#Results
print(ldamodel.print_topics(num_topics=1, num_words=4))
```

Packages Needed (Commands for a Linux System):

1. *sudo apt-get install python-tk*
2. *sudo apt-get install python-matplotlib*
3. *sudo pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose*
4. *sudo apt-get install python-dev libxml2-dev libxslt1-dev antiword unrtf poppler-utils pstotext tesseract-ocr flac ffmpeg lame libmad0 libsox-fmt-mp3 sox libjpeg-dev*
5. *sudo pip install textract*
6. *sudo pip install --upgrade gensim*

//Python provides many great libraries for text mining practices, “gensim” is one such clean and beautiful library to handle text data. It is scalable, robust and efficient.

7. Open a Python console and do the following:

>>> *import nltk*

>>> *nltk.download()*

Working:

```
gopal@gopal-virtual-machine ~/Desktop/Opinion Modelling/TextExtracter_new $ python main2.py
Enter the path of the pdf file (eg. C:/Games/abc.pdf)
2.pdf
[(0, u'0.044*"packet" + 0.034*"source" + 0.025*"switch" + 0.022*"switching"')]
gopal@gopal-virtual-machine ~/Desktop/Opinion Modelling/TextExtracter_new $
```

We can then deduce that the given pdf tells us about ‘Switching techniques’ or ‘Packet switching’.

Input: PDF FILE PATH

OUTPUT: Top 4 related words for that topic.

Result:

By using the topic modelling technique, we can get an insight so as to what a given document is all about and can tell about the theme of the document.