# Project Report:

# Topic Modeling using:
# Latent Dirichlet Allocation

**Submitted By:**
**Anurag Sharma**

**Under Guidance Of:**
**Dr. Shakir Bilal**
**Assistant Professor**
**AIISH**
**Amity University Haryana**


**Department of Data Science**
**Amity Institute of Integrity and Health**
**Amity University Haryana**

# Index

# Certificate

**This is certify that Mr. Anurag Sharma, a bonafide student of M.Sc. Data Science 3rd Semester, Amity Institute of Integrated Science and Health, Amity university Haryana has successfully completed the summer internship project in the area of "NLP" under the supervision of Dr. Shakir Bilal for the period July 2021 to August 2021 and submitted his project report with title "Latent Dirichlet Allocation" to department in the partial fulfilment of his degree. No part of the project submitted before any other degree.**

**Dr. Shakir Bilal**
**Assistant Professor**
**Amity Institute of Integrative Science and Health**
**Amity University, Haryana.**

# Acknowledgement

**I deem it pleasure to acknowledge our sense of gratitude to our project guide Dr. Shakir Bilal under whom I have carried out my project work. His incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work. I also wish to reciprocate in full measure the kindness shown by Dr. Shakir Bilal who inspired us with his valuable suggestions in successfully completing the project work.**

**Anurag Sharma**
**A525117720023**
**AIISH**
**Amity University, Haryana**

# Abstract

**Labeling the large number of documents based on the content they are containing. This helps to categorize them and then can be used for further study as per the need of the operations. The documents are labeled based on the amount of the topic proportion included in it. So we can know what percent of an individual topic is contributing or related to the corpora.**

# Introduction

**Topic modeling is in demand and will always be in demand as long as text exists and utilizes as a mode of communication. Topic modeling processes include several techniques to label data and are also known for their different uses. There are several methods like Latent Dirichlet Allocation, Correlation Topic Modeling, Structured Topic Modeling, Dynamic Topic Modeling and we are using LDA i.e., Latent Dirichlet Allocation. Topic Modeling assigns topic to the document in the corpus and utilization of this aspect can be implemented as per its need. Suppose my website contains blogs or articles or research papers so I can use topic modeling techniques in the search engine of my website. I can also manage the content of my website via topic modeling techniques.**

**In I.T there are so many fields they merges and become fewer when we eventually make a product comprises of many services in it eg. gadgets like mobile phones, laptops, smart TVs, etc. but even a website can contain or include nearly all parts of the I.T sector, from A.I applications to hardwares on which the applications are running.**

**And we all know classification is a very important part of A.I but accuracy is in demand.**

**If the data is in textual form and are needed to categorize then document form data can be categorized by topic modeling.**

**Similarly if the data is available in the form of image or video or audio then it can be categorized by using suitable algorithms plus methods respectively but this is out of the scope of this paper.**

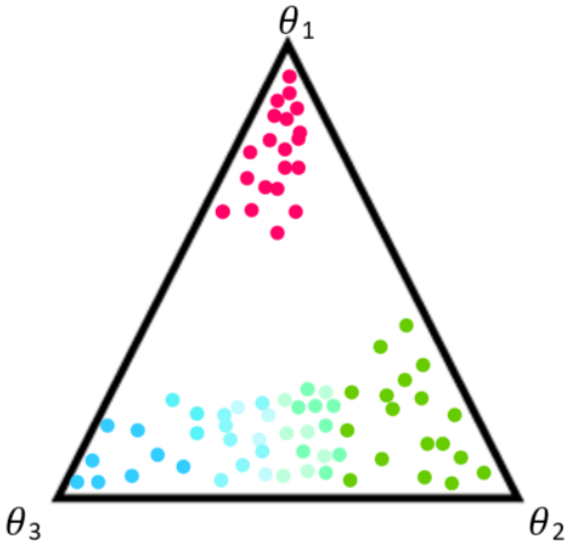
# Latent Dirichlet Allocation

**Latent Dirichlet Allocation Model is a generative probabilistic model i.e., it assigns topics on the basis of generated probabilities.**
**The documents are randomly arranged and each document is a mixture of equal or unequal number of words.**
**Words are represented as $w$ and a single document is represented as w**
**Let N be the number of words over a poisson distribution. And the topics which are going to be assigned are over dirichlet distribution.**

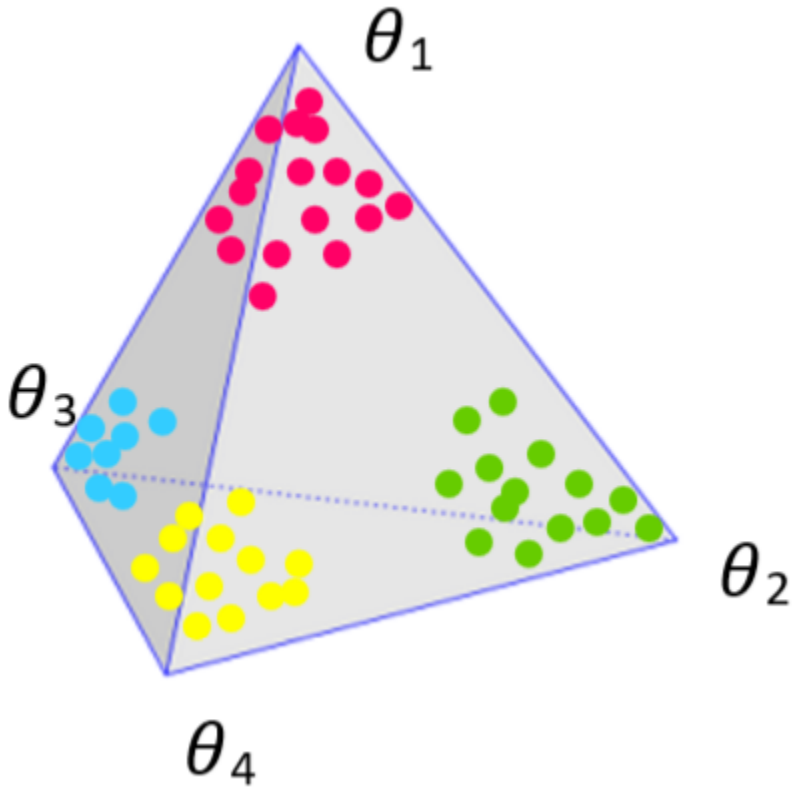
# Methodology

**We assume the following things, which have been used in the equations:-**

1. **"N" words in a Document ~ Poisson distribution.**
2. **Multinomial θ ~ Dirichlet distribution(α).**
3. **Topics z~θ.**
4. **Let a word $w_n$, multinomial probability conditioned on topic, $p(w| z_n,\beta)$.**



This is 2-dimensional simplex and three corners can be assumed as topics let say $\theta_1$, $\theta_2$, $\theta_3$. And the colourful dots represent how the words are distributed over these three topics.




This is 3-dimensional simplex which corner representing four topics i.e., $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$. And similarly these colourful dots are also representing the proportion of topics over them.

**NOTE: Each corner in the diagram will have equal distance from one another i.e., each topic in the simplex will maintain equal distance from one another.**

**The dimensional simplex can be assumed similarly and further an equation representation of it is:**

$$\left\{x \in \mathbb{R}^k : x_0 + \cdots + x_{k-1} = 1, x_i \geq 0 \text{ for } i = 0, \ldots, k-1\right\}.$$

This equation is for the k-1 dimensional simplex where having k topics over it's corners respectively. $x_0$, $x_1$, $x_2$,..$x_{k-1}$ are the proportion or probabilities over k topics which sum equals to 1.

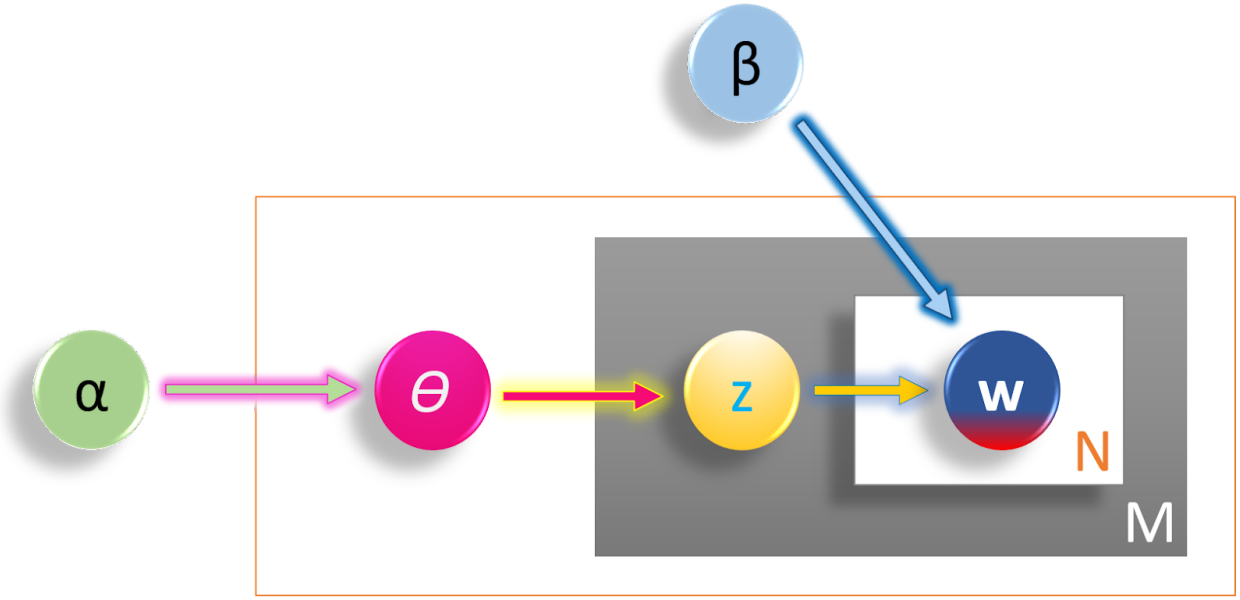
**NOTE: $x_0$ is same as $\theta_1$ and similarly other values are also. So $\theta_1 + \theta_2 + \theta_3 + ... + \theta_k = 1$.**

**If this θ is drawn from dirichlet distribution for k-1 dimensional simplex then the probability density function is given by:**

$$p(\theta|\alpha) = \frac{\Gamma\left(\sum_{i=1}^{k}\alpha_i\right)}{\prod_{i=1}^{k}\Gamma(\alpha_i)}\theta_1^{\alpha_1-1}\cdots\theta_k^{\alpha_k-1},$$

# Architecture

## LDA model diagram:




**As shown in the diagram θ is drawn from dirichlet distribution and z topics is drawn from θ.**

**Marginal Probability over the document is given by:**

$$p(\mathbf{w}|\alpha,\beta) = \int p(\theta|\alpha)\left(\prod_{n=1}^{N}\sum_{z_n} p(z_n|\theta)p(w_n|z_n,\beta)\right)d\theta.$$

**Probability of corpus is given by: (product over all probabilities of documents).**
**NOTE: There are total M documents. Whereas D represents the corpus.**

$$p(\mathcal{D}|\alpha,\beta) = \prod_{d=1}^{M}\int p(\theta_d|\alpha)\left(\prod_{n=1}^{N_d}\sum_{z_{dn}} p(z_{dn}|\theta_d)p(w_{dn}|z_{dn},\beta)\right)d\theta_d.$$

**Based on De Finetti's : The topics are infinitely exchangeable in the document.**

$$p(z_1,\ldots,z_N) = p(z_{\pi(1)},\ldots,z_{\pi(N)}).$$

## Practical Implementation:

**In this project the model is going to be trained over "wikipedia" data. Simply impeccable implementation of wikipedia (module: wikipedia) and Wikipedia-API (module: wikipediaapi) libraries make it easier to retrieve the data from the wikipedia server.**
**Modules provide various functions like search, page and attributes like content, text, url, summary, etc. which is helpful for collecting data from wikipedia.**
**Step1: Search the query, collect the resulting titles and pull the full-text data of the page and make a corpus.**
**These are the search queries on wikipedia web server.**

```
quries = ["electron", "proton", "neutron", "fairy tails", "computer science", "jokes", "new"]
```

**We have collected 15 results of each term i.e., 105 documents with its title.**
**Example of search "Electron" for 15 results we have titles:**

```
['Electron',
 'Electron microscope',
 'Electron transport chain',
 'Scanning electron microscope',
 'Valence electron',
 'Electron configuration',
 'Electron ionization',
 'Electron shell',
 'Electron diffraction',
 'Electron affinity',
 'Electronvolt',
 'Transmission electron microscopy',
 'Covalent bond',
 'Electron (disambiguation)',
 'Electron capture']
```

**Step2: Using nltk library the data preprocessing operations take place,like tokenizing,lowering,lemmatization, etc.**

**Step3: I have used stopwords in this step whereas some other tutorials are using this step later but it depends on the requirement (as I found here of this process).**
**The below result is shown self-compiled but just for example because the length of it is so, random cut-out is shown.**

**Step4: Here we are using the gensim library for finding out bigrams (we also find out trigrams or ngrams).**

**Left side is the bigram and the right side from which document it is generated i.e., document name.**

```
new_statesman ============== New Statesman
new_statesman ============== New Statesman
new_statesman ============== New Statesman
new_delhi ============== New Delhi
national_capital ============== New Delhi
territory_delhi ============== New Delhi
new_delhi ============== New Delhi
government_india ============== New Delhi
supreme_court ============== New Delhi
new_delhi ============== New Delhi
municipal_council ============== New Delhi
new_delhi ============== New Delhi
national_capital ============== New Delhi
territory_delhi ============== New Delhi
new_delhi ============== New Delhi
relatively_small ============== New Delhi
national_capital ============== New Delhi
much_larger ============== New Delhi
new_delhi ============== New Delhi
wa_inaugurated ============== New Delhi
nineteenth_century ============== New Delhi
new_delhi ============== New Delhi
wa_officially ============== New Delhi
east_coast ============== New Delhi
three_day ============== New Delhi
new delhi ============== New Delhi
```

```
late_19th ============== New Caledonia
new_caledonia ============== New Caledonia
ethnic_group ============== New Caledonia
new_caledonia ============== New Caledonia
grande_terre ============== New Caledonia
roman_catholic ============== New Caledonia
new_caledonia ============== New Caledonia
loyalty_island ============== New Caledonia
late_20th ============== New Caledonia
new_caledonia ============== New Caledonia
new_caledonia ============== New Caledonia
roman_catholic ============== New Caledonia
new_caledonia ============== New Caledonia
noum_accord ============== New Caledonia
secondary_education ============== New Caledonia
new_caledonia ============== New Caledonia
high_school ============== New Caledonia
secondary_education ============== New Caledonia
new_caledonia ============== New Caledonia
new_caledonia ============== New Caledonia
nouvelle_cal ============== New Caledonia
```

**Step5: In this step with the gensim library we will make a bag of words by dictionary module.**

```
Dictionary(2060 unique tokens: ['ability', 'able', 'absolute', 'absorbed', 'absorption']...)
```

## Step6: Now we will train our model by gensim.models.LdaModel(<parameters>).

```
INFO : using autotuned alpha, starting with [0.14285715, 0.14285715, 0.14285715, 0.14285715, 0.14285715, 0.14285
INFO : using serial LDA version on this node
INFO : running online (multi-pass) LDA training, 7 topics, 30 passes over the supplied corpus of 105 documents, u
INFO : -8.226 per-word bound, 299.5 perplexity estimate based on a held-out corpus of 105 documents with 125136 w
INFO : PROGRESS: pass 0, at document #105/105
INFO : optimized alpha [0.05146245, 0.08153991, 0.06892042, 0.069553025, 0.03244009, 0.111964785, 0.063808]
INFO : topic #4 (0.032): 0.019*"electron" + 0.015*"joke" + 0.009*"russian" + 0.007*"energy" + 0.007*"fairy" + 0.0
INFO : topic #0 (0.051): 0.012*"computer" + 0.011*"data" + 0.009*"language" + 0.007*"team" + 0.007*"type" + 0.00
INFO : topic #3 (0.070): 0.028*"neutron" + 0.020*"age" + 0.013*"star" + 0.007*"energy" + 0.006*"nuclear" + 0.005
INFO : topic #1 (0.082): 0.013*"proton" + 0.011*"season" + 0.008*"show" + 0.008*"series" + 0.008*"deal" + 0.007*
INFO : topic #5 (0.112): 0.031*"electron" + 0.014*"proton" + 0.013*"computer" + 0.008*"protein" + 0.008*"joke" +
INFO : topic diff=1.089404, rho=1.000000
INFO : -7.061 per-word bound, 133.6 perplexity estimate based on a held-out corpus of 105 documents with 125136 w
INFO : PROGRESS: pass 1, at document #105/105
INFO : optimized alpha [0.04431092, 0.07559344, 0.055657543, 0.052604828, 0.029156797, 0.09252877, 0.052634142]
INFO : topic #4 (0.029): 0.022*"russian" + 0.019*"joke" + 0.012*"fairy" + 0.011*"electron" + 0.011*"soviet" + 0.0
INFO : topic #0 (0.044): 0.018*"computer" + 0.016*"data" + 0.013*"language" + 0.009*"system" + 0.009*"type" + 0.0
INFO : topic #2 (0.056): 0.014*"city" + 0.009*"england" + 0.007*"proton" + 0.007*"american" + 0.007*"area" + 0.00

INFO : PROGRESS: pass 28, at document #105/105
INFO : optimized alpha [0.031909246, 0.07143154, 0.05010368, 0.03160029, 0.03636265, 0.04937651, 0.030662047]
INFO : topic #6 (0.031): 0.059*"joke" + 0.012*"text" + 0.011*"book" + 0.010*"century" + 0.009*"study" + 0.008*"form" + 0
INFO : topic #3 (0.032): 0.063*"neutron" + 0.037*"age" + 0.024*"star" + 0.014*"nuclear" + 0.013*"energy" + 0.013*"react
INFO : topic #5 (0.049): 0.057*"electron" + 0.015*"proton" + 0.013*"energy" + 0.011*"protein" + 0.011*"atom" + 0.010*"be
INFO : topic #2 (0.050): 0.015*"city" + 0.010*"american" + 0.009*"england" + 0.008*"government" + 0.008*"deal" + 0.007*
INFO : topic #1 (0.071): 0.030*"proton" + 0.015*"season" + 0.013*"show" + 0.011*"episode" + 0.010*"series" + 0.008*"tea
INFO : topic diff=0.017479, rho=0.182574
INFO : -6.779 per-word bound, 109.8 perplexity estimate based on a held-out corpus of 105 documents with 125136 words
INFO : PROGRESS: pass 29, at document #105/105
INFO : optimized alpha [0.032075793, 0.07164686, 0.050430227, 0.03158166, 0.036709435, 0.049302034, 0.030803168]
INFO : topic #6 (0.031): 0.058*"joke" + 0.012*"text" + 0.011*"book" + 0.010*"century" + 0.009*"study" + 0.008*"form" + 0
INFO : topic #3 (0.032): 0.063*"neutron" + 0.037*"age" + 0.024*"star" + 0.014*"nuclear" + 0.013*"energy" + 0.013*"react
INFO : topic #5 (0.049): 0.057*"electron" + 0.015*"proton" + 0.013*"energy" + 0.011*"protein" + 0.011*"atom" + 0.010*"be
INFO : topic #2 (0.050): 0.015*"city" + 0.010*"american" + 0.009*"england" + 0.008*"government" + 0.008*"deal" + 0.007*
INFO : topic #1 (0.072): 0.030*"proton" + 0.015*"season" + 0.013*"show" + 0.011*"episode" + 0.010*"series" + 0.008*"tea
INFO : topic diff=0.016508, rho=0.179605
```

# Result

**These are the proportion of the specific words which are present in the result with probability distributions and represent that which one tell us more about the document. Or we can say generated topics from the applied algorithms.**

```
[(0,
  '0.039*"computer" + 0.020*"data" + 0.018*"science" + 0.014*"system" +
0.014*"language" + 0.012*"object" + 0.012*"programming" +
0.011*"computer_science" + 0.011*"software" + 0.009*"program"'),
 (1,
  '0.030*"proton" + 0.015*"season" + 0.013*"show" + 0.011*"episode" +
0.010*"series" + 0.008*"team" + 0.007*"character" + 0.007*"game" +
0.006*"car" + 0.006*"later"'),
 (2,
```

```
  '0.015*"city" + 0.010*"american" + 0.009*"england" + 0.008*"government"
+ 0.008*"deal" + 0.007*"population" + 0.007*"area" + 0.006*"united" +
0.006*"university" + 0.006*"french"'),
 (3,
  '0.063*"neutron" + 0.037*"age" + 0.024*"star" + 0.014*"nuclear" +
0.013*"energy" + 0.013*"reaction" + 0.010*"mass" + 0.010*"music" +
0.008*"nucleus" + 0.007*"movement"'),
 (4,
  '0.022*"russian" + 0.021*"joke" + 0.016*"fairy" + 0.013*"soviet" +
0.011*"ring" + 0.010*"say" + 0.007*"man" + 0.007*"get" + 0.006*"back" +
0.006*"american"'),
 (5,
  '0.057*"electron" + 0.015*"proton" + 0.013*"energy" + 0.011*"protein" +
0.011*"atom" + 0.010*"beam" + 0.009*"sample" + 0.008*"particle" +
0.007*"structure" + 0.007*"image"'),
 (6,
  '0.058*"joke" + 0.012*"text" + 0.011*"book" + 0.010*"century" +
0.009*"study" + 0.008*"form" + 0.007*"social" + 0.007*"word" +
0.006*"system" + 0.006*"theory"')]
```

**Now we will compare the nature of query and the result we have get back from the algorithms.**

**result back from the algorithms: (let say R1)**

```
['city', 'computer', 'joke', 'electron', 'russian', 'neutron', 'proton']
```

**nature of query: (let say Q1)**

```
['electron', 'proton', 'neutron', 'fairy tails', 'computer science',
'jokes', 'new']
```

**We can see by this, nearly almost results are similar but “russian” and “city” in R1 not matching with any of the searched query. And also “fairy tails” and “new” doesn’t have any similar result in R1.**
**We can find out the difference just by taking an insight of it.**
**When we query like “electron” or “computer science” we get result like:**

**For “electron”:**

```
['Electron',
 'Electron microscope',
 'Electron transport chain',
 'Scanning electron microscope',
 'Valence electron',
 'Electron configuration',
 'Electron ionization',
 'Electron shell',
 'Electron diffraction',
 'Electron affinity',
 'Electronvolt',
 'Transmission electron microscopy',
 'Covalent bond',
 'Electron (disambiguation)',
 'Electron capture']
```

**For “computer science”:**

```
['Computer science',
 'Computer science and engineering',
 'Glossary of computer science',
 'Abstraction (computer science)',
 'Computer graphics (computer science)',
 'Computer science education',
 'Record (computer science)',
 'Semantics (computer science)',
 'Heuristic (computer science)',
 'History of computer science',
 'AP Computer Science',
 'Bachelor of Computer Science',
 'Theoretical computer science',
 'Object (computer science)',
 'Polling (computer science)']
```

**From here we can know that each resulted title is related to searched query so it means the chances of having this word or this kind of words is more. It can be in combinations like bigrams and trigrams (or can be in ngrams). The LDA model is generally for quantitative analysis. We can generate topics by repetitive appearance of a particular word or bigrams or n-grams in a document or corpus.**

**For “fairy tails”:**

```
['List of My Little Pony characters',
 'Pennisetum',
```

```
 'Mickey Maguire (Shameless)',
 'Ian Gallagher',
 'The Boxcar Children (film)',
 'Fairy ring',
 'Pembroke Welsh Corgi',
 'Jazz Jennings',
 'Puppy Dog Pals',
 'Mio Shirai',
 'Fox spirit',
 'Adventures of Sonic the Hedgehog',
 'Donkey (Shrek)',
 'Kumiho',
 'Far darrig']
```

For “new”:

```
 ['New',
  'New Statesman',
  'New Delhi',
  'New Caledonia',
  'New Testament',
  'New Age',
  'New Jersey',
  'New England',
  'New Brunswick',
  'New Deal',
  'New Girl',
  'New wave music',
  'New Orleans',
  "New Year's Six",
  'New Balance']
```

If we notice the search result of the “fairy tails” the nature of the query doesn’t reflex on the resulted titles,as we are doing quantitative analysis so our model will result like popular name or destination or other fantasy words embedded in the story as words like “russian” and “city” is shown in the result.
And also if we see the query “new” results title which contain keyword new possibly the content is about it’s partner words like Statesman, Age, etc. and new is tends to like an adjective so the content covers more information about its partner words which are sharing the title.

**So eventually we can say if we are going to apply LDA then our query terms will also be able to determine the type of content for this problem.**

**Draw back of LDA**

**The Drawback of LDA is it doesn't matter if similar kinds of words are there in the document, the major count of the top words is eventually going to affect the result.**

**Example: list like:-**

**["sad","sad","sad","good","best","happiest","fantabulous","superb","amazing"]**

**It is going to suggest "sad" with more probability. Some other models can handle these features to some extent like CTM Correlation Topic Modeling, but simple implementation CTM is not suggested for large scale of data, while LDA can be implemented on large scale of data.**