Latent Dirichlet Allocation

Harsh Bansal · Follow

Published in Analytics Vidhya · 8 min read · Mar 3, 2020

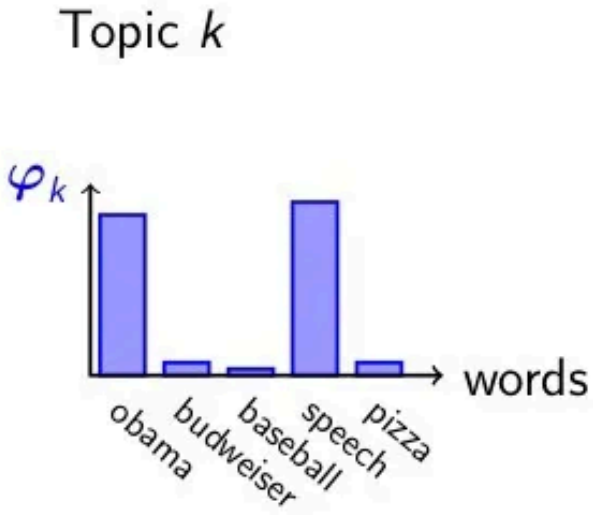
Latent Dirichlet Allocation (LDA) serves as a topic modeling technique, adept at categorizing text within a document into specific topics. It leverages the Dirichlet distribution to discern topics for each document model and words for each topic model.

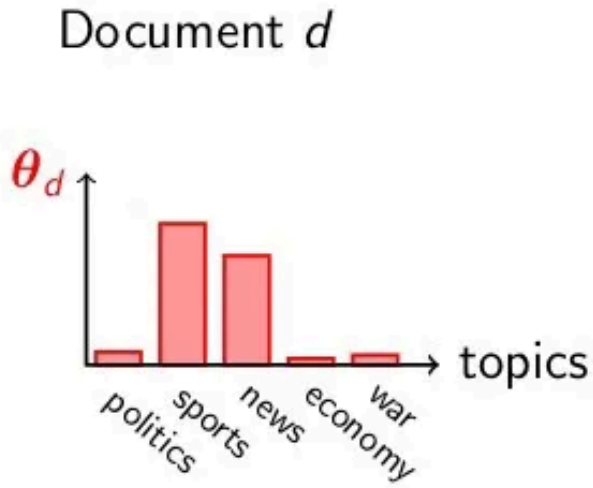
Johann Peter Gustav Lejeune Dirichlet, a prominent German mathematician of the 19th century, made significant contributions to the advancement of modern mathematics. Among his enduring legacies is the Dirichlet Distribution, a probability distribution named in his honor, which serves as the foundation of Latent Dirichlet Allocation (LDA).

Latent Dirichlet Allocation

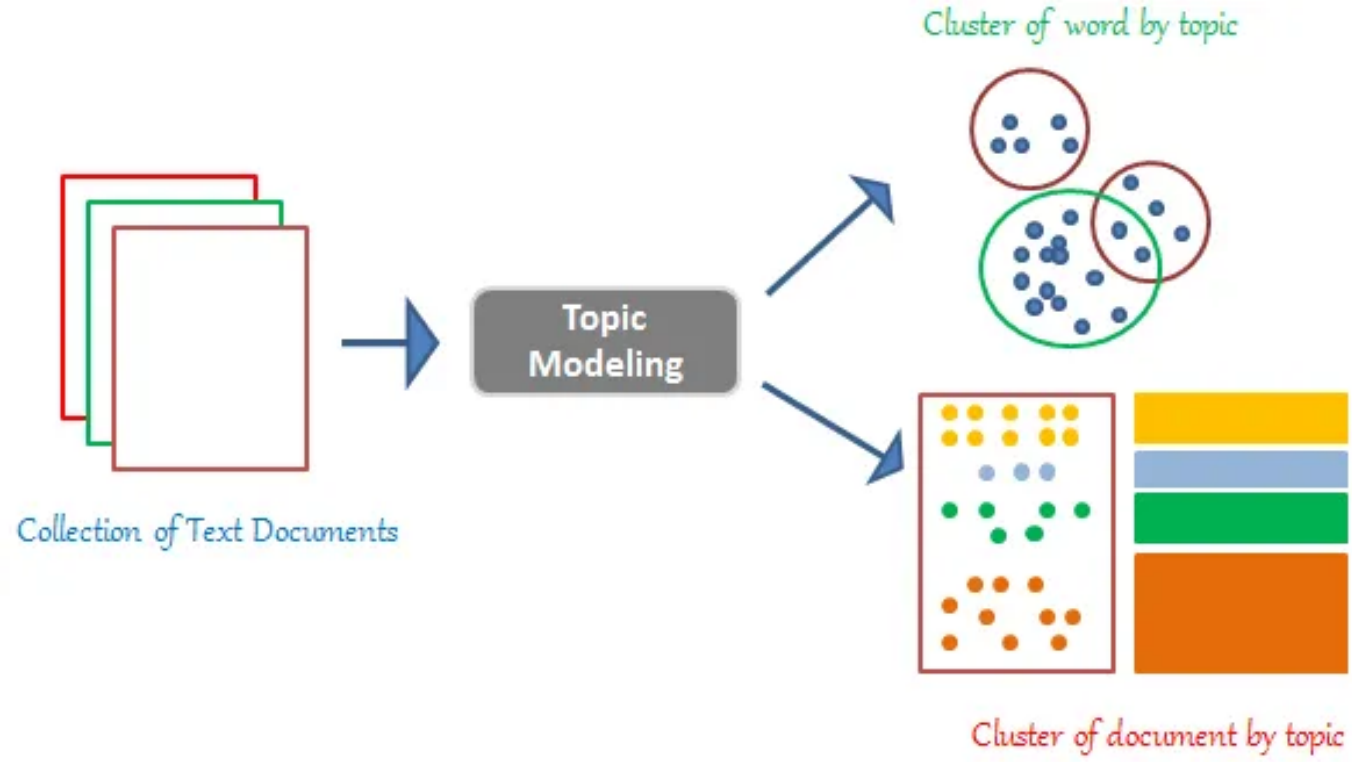
LDA discovers topics into a collection of documents.

LDA tags each document with topics.





How LDA works?

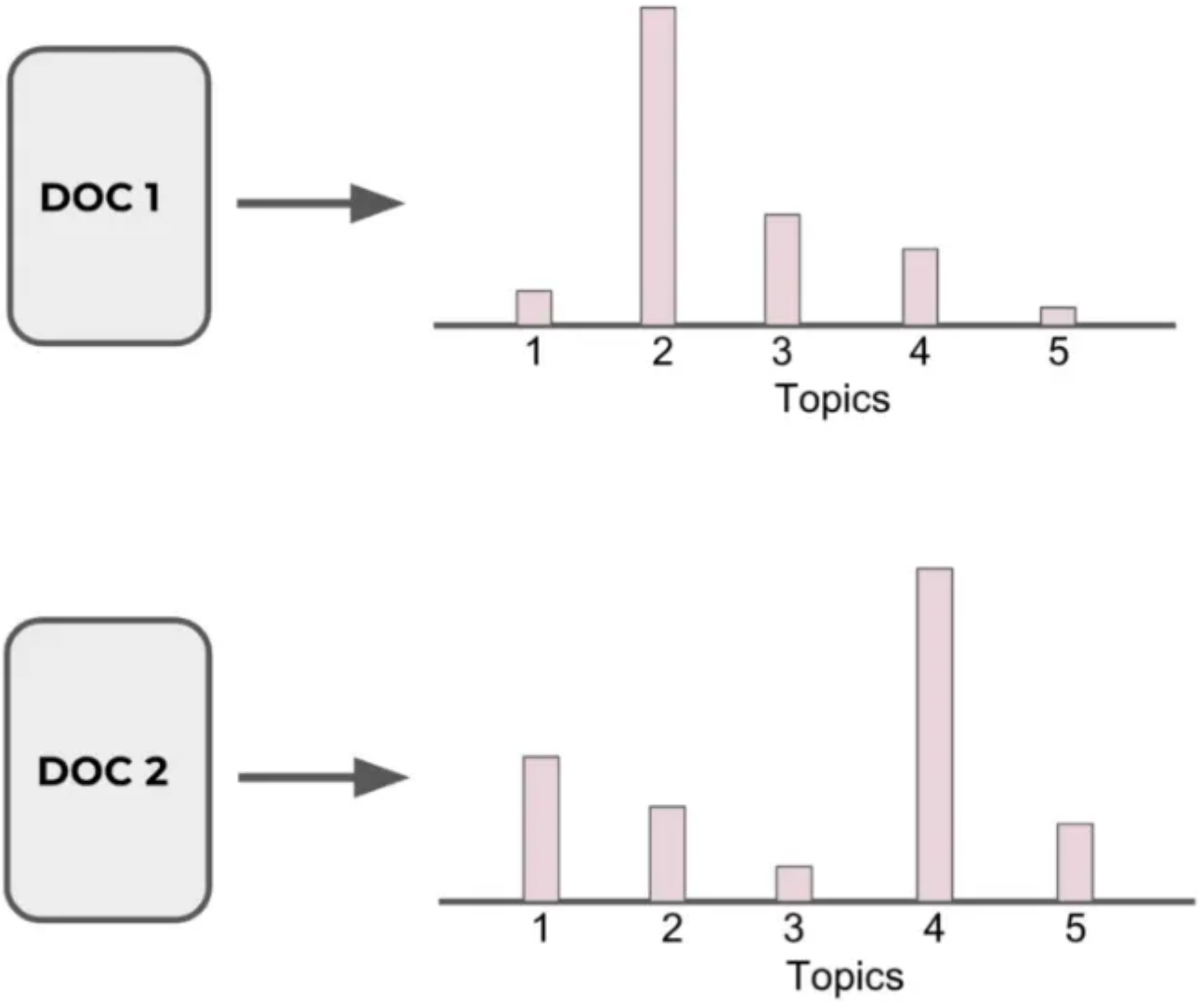


Latent Dirichlet Allocation (LDA) is a method for associating sentences with topics. LDA discerns specific topic sets based on the topics provided to it. Prior to generating these topics, LDA undergoes several preparatory processes. We establish a set of rules and facts before applying these processes.

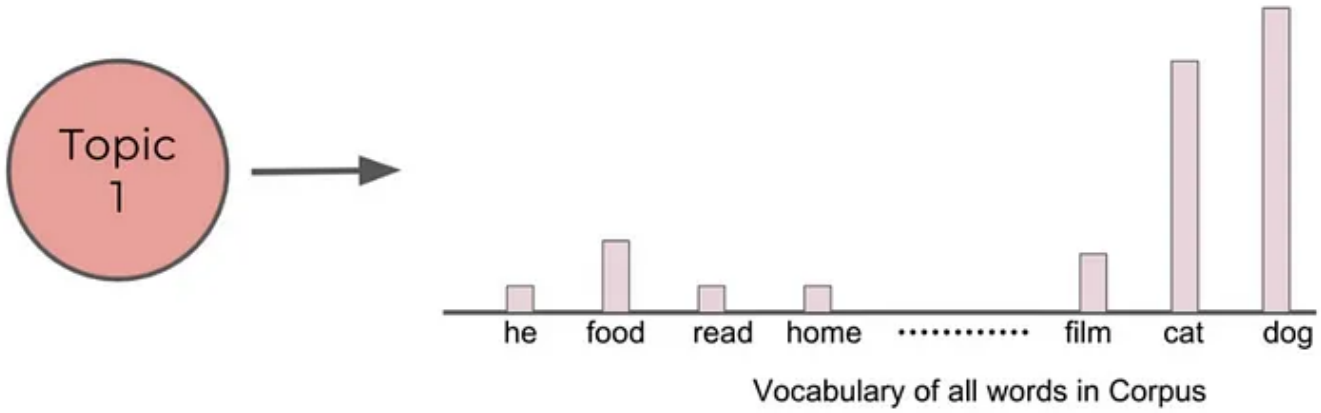
Assumptions of LDA for Topic Modelling:

- Documents with similar topics use similar groups of words
- Latent topics can then be found by searching for groups of words that frequently occur together in documents across the corpus

- Documents can be viewed as probability distributions across latent topics, indicating that certain documents will contain a higher proportion of words related to specific topics.



- Topics themselves are probability distribution over words



These are the assumptions users must understand before applying LDA.

Explanation through example:

Suppose we have the following statements:

- Cristiano Ronaldo and Lionel Messi are both great players of football
- People also admire Neymar and Ramos for their football skills
- The USA and China both are powerful countries
- China is building the largest air purifier
- India is also emerging as one of the most developing country by promoting football a global scale

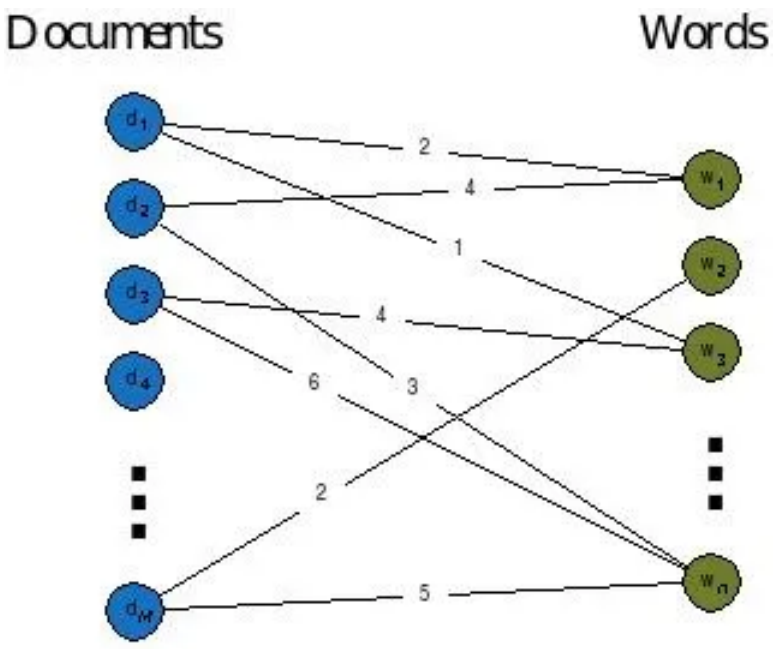
With the help of LDA, we can generate sets of topics about which the sentences are. If we take 2 topic sets into consideration then:

- Sentence 1 and Sentence 2 both belong to topic 1

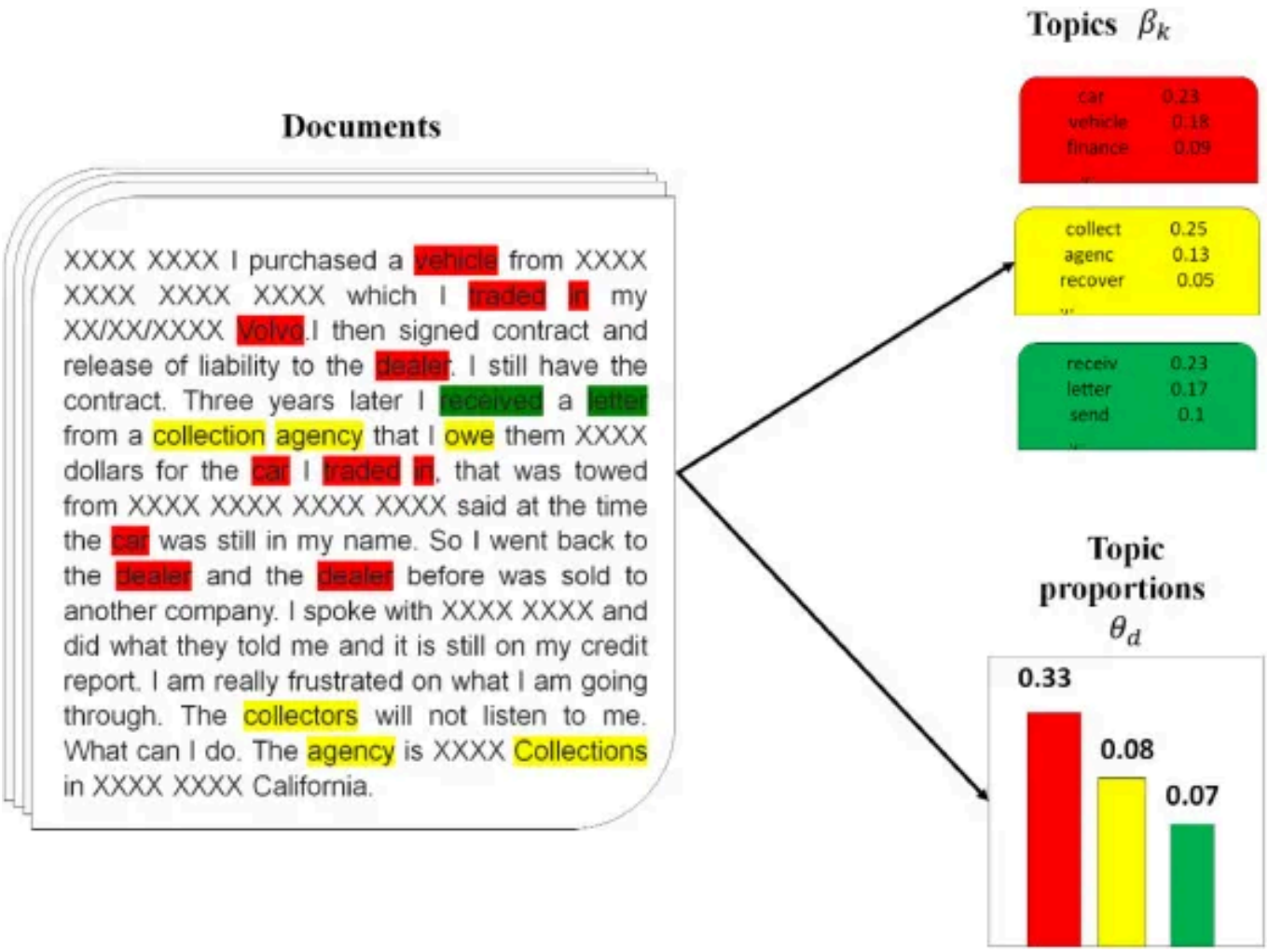
LDA, or Latent Dirichlet Allocation, posits that each document comprises a variety of contexts that correspond to different topics. Consequently, a document can be depicted as a collection of diverse topics, each composed of numerous words with specific probabilities. In LDA, each document possesses its unique characteristics, and the model assumes certain rules and regulations before generating a document. For instance, there is typically a word limit imposed, with users specifying a certain amount of words. Additionally, diversity in document content is encouraged, with documents ideally referencing a range of contexts, such as 60% business, 20% politics, and 10% food. Moreover, every keyword within a document is associated with a particular topic, and this relationship can be inferred using multinomial distribution. For instance, a word associated with the business domain might have a probability of occurrence at

3/5, while its association with politics could be at 1/5, as we discussed previously

Assuming this model is applied to a collection of documents, LDA endeavors to backtrack from the documents to identify a set of topics that are relevant to the context of the documents.



Now we try to understand its full working

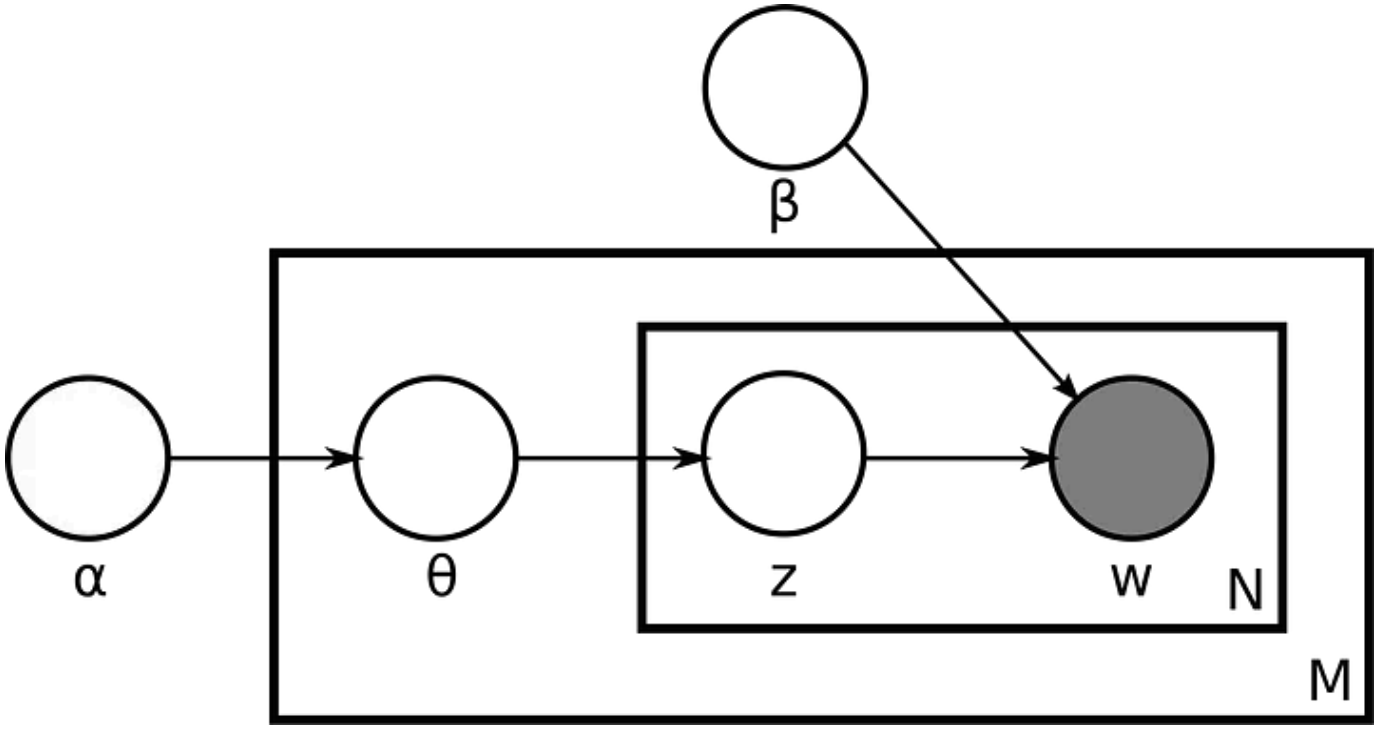


Since we have a set of documents sourced from a specific dataset or obtained randomly, we first determine a fixed number of K topics to uncover. We then employ LDA to learn the topic representation of each document and the words associated with each topic.

The LDA algorithm proceeds by iterating through each document and randomly assigning each word in the document to one of the K topics. This initial random assignment provides initial topic representations for all documents and word distributions for all topics. Subsequently, LDA iterates over

every word in every document to refine these topics. However, the initial topic representations may not be optimal, necessitating improvement. To address this limitation, a formula is devised where the core functionality of LDA comes into play.

Plate Notation representing LDA model:



M denotes the number of documents
N is the number of words in a given document (document i has {\displaystyle N_{i}}N_{i} words)
α is the parameter of the Dirichlet prior on the per-document topic distributions
β is the parameter of the Dirichlet prior on the per-topic word

distribution
theta is the topic distribution for document i
varphi is the word distribution for topic k
z is the topic for the j-th word in document i
w is the specific word.

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\beta_i) \prod_{i=1}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p(z_{d,n}|\theta_d)\, p(w_{d,n}|\beta_{1:K}, z_{d,n}) \right)$$

Explanation with simple terms:

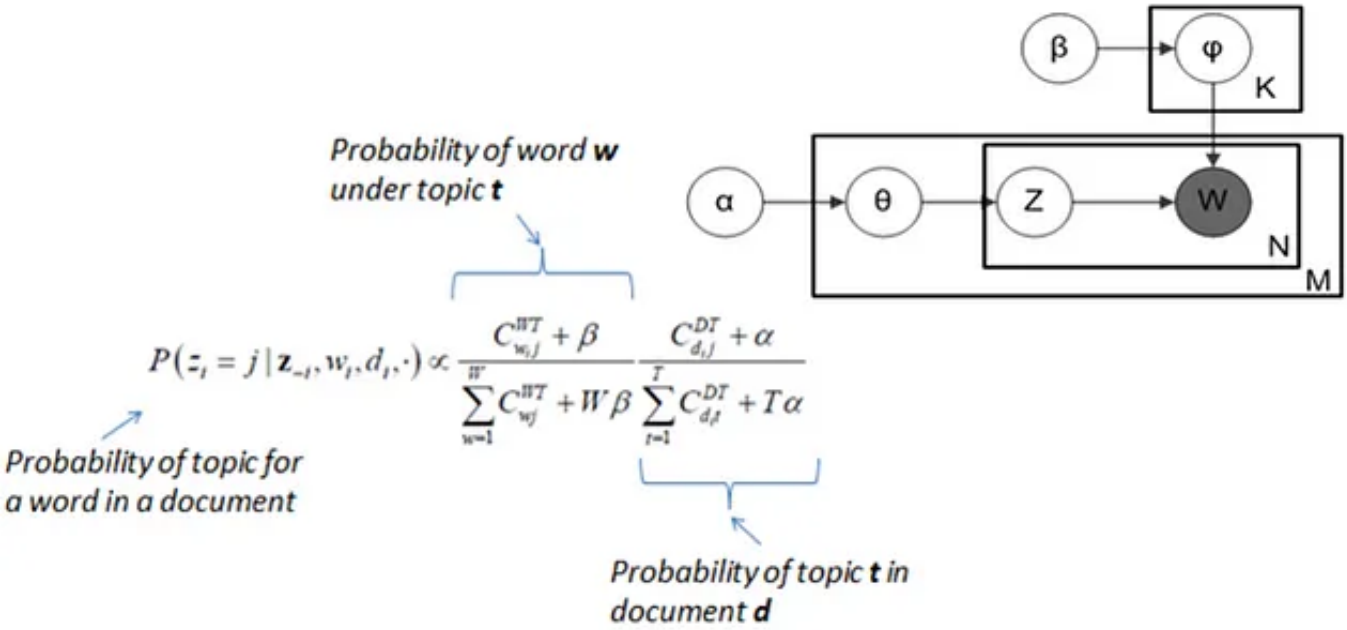
For every word in every document and for each Topic T, we calculate:
P(Topic T | Document D) = the proportion of words in Document D that are currently assign to Topic T

P(Word W | Topic T) = the proportion of assignments to topic T over all documents that come from this word W

Reassign w to a new topic where we choose Topic T with probability **P(Topic T | Document D) * P(Word W | Topic T)**.This is essentially that Topic T generated word w

After repeating the previous step a large number of times, we eventually reach a roughly steady state where the assignments

become acceptable. At this stage, each document is assigned to a specific topic. We can then search for the words that have the highest probability of being assigned to a given topic.



We ended up output such as

- Document assigned to topic 4
- Most common words (highest probability) for topic 4 (‘cat’,’vet’,’birds’,’dog’…)
- It is up to the user to interpret these topics.

Two important notes:

- The users must decide on amount of topics present in the document
- the user must interpret what the topics are

In essence, with LDA, if we possess a collection of documents and aim to generate a set of topics for representing these documents, we can achieve this through LDA. The process involves training LDA on each document, wherein it iterates through the document and assigns words to topics. However, this isn't a single-step process. Initially, LDA randomly assigns words to topics, kicking off a learning procedure. It then traverses through each word in each document, applying the formula discussed earlier. Through numerous iterations of this procedure, it eventually yields a set of topics.

Implementation

We will try to understand LDA more briefly by applying it to a dataset.

The dataset we're utilizing comprises information or news sourced from www.npr.org, encompassing the latest global news. Our objective is to implement LDA on these news columns to discern the most prevalent topics worldwide. Additionally, we

aim to assign topics to future news articles based on our findings.

Data Pre-Processing:

```
import pandas as pd
df = pd.read_csv('npr.csv')
df.head()
```

	Article
0	In the Washington of 2016, even when the polic...
1	Donald Trump has used Twitter — his prefe...
2	Donald Trump is unabashedly praising Russian...
3	Updated at 2:50 p. m. ET, Russian President Vl...
4	From photography, illustration and video, to d...

Notice how we don't have the topic of the articles! Let's use LDA to attempt to figure out clusters of the articles.

```
from sklearn.feature_extraction.text import
CountVectorizer
cv = CountVectorizer(max_df=0.95, min_df=2,
stop_words='english')
dtm = cv.fit_transform(df['Article'])
```

Count Vectorizer: CountVectorizer is a fundamental component of natural language processing and is often paired with TF-IDF. However, in our case, we opt for TF-IDF over CountVectorizer. TF-IDF calculates the frequency of a token's occurrence in a document and utilizes this value as its weight. We utilized CountVectorizer to convert the text data into a machine-readable format.

- **max_df:** float in the range [0.0, 1.0] or int, default=1.0
  It is used to remove words that appear too frequently. If max_df = 0.50 means "ignore terms that appear in more than 50% of the documents". If max_df = 25 means "ignore terms that appear in more than 25 documents". The default max_df is 1.0, which means "ignore terms that appear in more than 100% of the documents". Thus, the default setting does not ignore any terms.
- **min_df:** float in range [0.0, 1.0] or int, default=1
  It is used to remove words that appear rarely. If min_df = 0.01 means "ignore terms that appear in less than 1% of the documents".If min_df = 5 means "ignore terms that appear in less than 5 documents". The default min_df is 1, which means "ignore terms that appear in less than 1 document". Thus, the default setting does not ignore any terms.

LDA model:

```
from sklearn.decomposition import
LatentDirichletAllocation
LDA =
LatentDirichletAllocation(n_components=7,random_state=42)
LDA.fit(dtm)
```

Showing Stored Words:

```
len(cv.get_feature_names())
>>>54777

for i in range(10):
    random_word_id = random.randint(0,54776)
    print(cv.get_feature_names()[random_word_id])

>>>cred
fairly
occupational
temer
tamil
closest
condone
breathes
tendrils
pivot

for i in range(10):
    random_word_id = random.randint(0,54776)
    print(cv.get_feature_names()[random_word_id])

>>>foremothers
mocoa
ellroy
```

```
liron
ally
discouraged
utterance
provo
videgaray
archivist
```

Showing top words per topic

```
len(LDA.components_)
>>>7
len(LDA.components_[0])
>>>54777

single_topic = LDA.components_[0]

# Returns the indices that would sort this array.
single_topic.argsort()

# Word least representative of this topic
single_topic[18302]

# Word most representative of this topic
single_topic[42993]

# Top 10 words for this topic:
single_topic.argsort()[-10:]
>>>array([33390, 36310, 21228, 10425, 31464,  8149,
36283, 22673, 42561,
       42993], dtype=int64)

top_word_indices = single_topic.argsort()[-10:]
for index in top_word_indices:
    print(cv.get_feature_names()[index])
```

These look like business articles perhaps. We will perform .transform() on our vectorized articles to attach a label number. But before, we view all the topics found.

```
for index,topic in enumerate(LDA.components_):
    print(f'THE TOP 15 WORDS FOR TOPIC #{index}')
    print([cv.get_feature_names()[i] for i in
topic.argsort()[-15:]])
    print('\n')
```

```
THE TOP 15 WORDS FOR TOPIC #0
['companies', 'money', 'year', 'federal', '000', 'new', 'percent', 'government', 'company', 'million', 'care', 'people', 'healt
h', 'said', 'says']


THE TOP 15 WORDS FOR TOPIC #1
['military', 'house', 'security', 'russia', 'government', 'npr', 'reports', 'says', 'news', 'people', 'told', 'police', 'presid
ent', 'trump', 'said']


THE TOP 15 WORDS FOR TOPIC #2
['way', 'world', 'family', 'home', 'day', 'time', 'water', 'city', 'new', 'years', 'food', 'just', 'people', 'like', 'says']


THE TOP 15 WORDS FOR TOPIC #3
['time', 'new', 'don', 'years', 'medical', 'disease', 'patients', 'just', 'children', 'study', 'like', 'women', 'health', 'peop
le', 'says']


THE TOP 15 WORDS FOR TOPIC #4
['voters', 'vote', 'election', 'party', 'new', 'obama', 'court', 'republican', 'campaign', 'people', 'state', 'president', 'cli
nton', 'said', 'trump']


THE TOP 15 WORDS FOR TOPIC #5
['years', 'going', 've', 'life', 'don', 'new', 'way', 'music', 'really', 'time', 'know', 'think', 'people', 'just', 'like']


THE TOP 15 WORDS FOR TOPIC #6
['student', 'years', 'data', 'science', 'university', 'people', 'time', 'schools', 'just', 'education', 'new', 'like', 'student
s', 'school', 'says']
```

Attaching Discovered Topic Labels to Original Articles

```
topic_results = LDA.transform(dtm)
npr['Topic'] = topic_results.argmax(axis=1)
```

	Article	Topic
0	In the Washington of 2016, even when the polic...	1
1	Donald Trump has used Twitter — his prefe...	1
2	Donald Trump is unabashedly praising Russian...	1
3	Updated at 2:50 p. m. ET, Russian President Vl...	1
4	From photography, illustration and video, to d...	2
5	I did not want to join yoga class. I hated tho...	3
6	With a who has publicly supported the debunk...	3
7	I was standing by the airport exit, debating w...	2
8	If movies were trying to be more realistic, pe...	3
9	Eighteen years ago, on New Year's Eve, David F...	2

Limitations

- There is a limit to the amount of topics we can generate
- LDA is unable to depict correlations which led to the occurrence of uncorrelated topics
- There is no development of topics over time
- LDA assumes words are exchangeable, sentence structure is not modeled

- Unsupervised (sometimes weak supervision is desirable, e.g. in sentiment analysis)

With this, you have the complete idea of Latent Dirichlet Allocation (LDA). Enjoy.

Machine Learning | Artificial Intelligence | Naturallanguageprocessing | Algorithms | Text Analysis

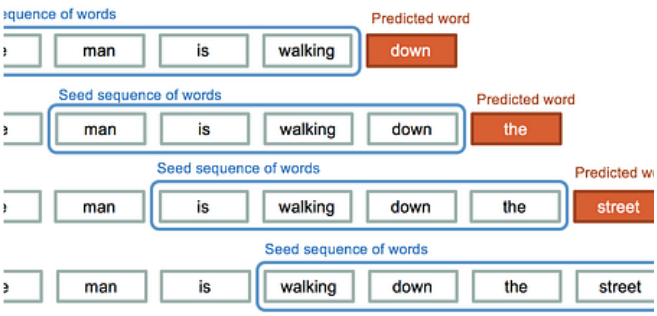
Written by Harsh Bansal

28 Followers · Writer for Analytics Vidhya

Data Scientist at Paytm

Follow

More from Harsh Bansal and Analytics Vidhya



Harsh Bansal

Text Generation Using LSTM

In text generation, we try to predict the next character or word of the...

Jul 17, 2020 35 2

Kia Eisinga in Analytics Vidhya

How to create a Python library

Ever wanted to create a Python library, albeit for your team at work or for...

Jan 26, 2020 2.6K 28

Harikrishnan N B in Analytics Vidhya

Confusion Matrix, Accuracy, Precision, Recall, F1 Score

Binary Classification Metric

Harsh Bansal in Analytics Vidhya

Credit Card Fraud Detection

A detailed comparison of neural networks and anomaly detection...

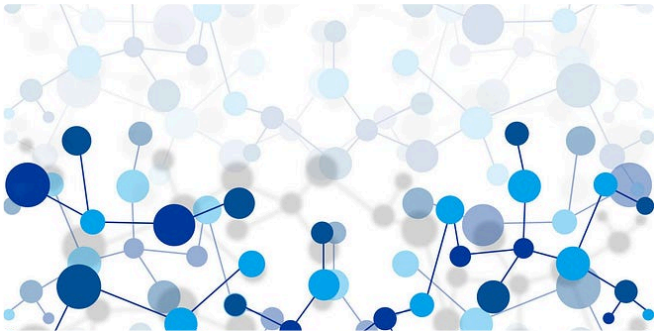
Dec 10, 2019 1K 6

Oct 16, 2019 39 2

See all from Harsh Bansal

See all from Analytics Vidhya

Recommended from Medium

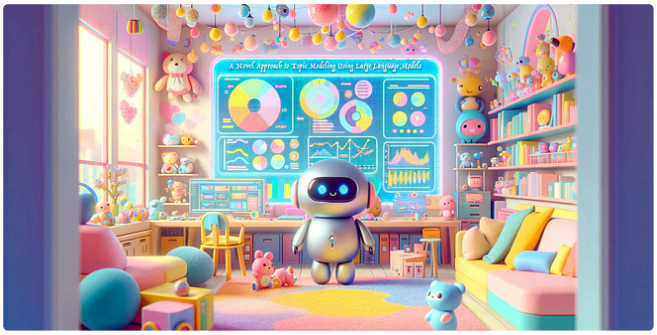


Petr Korab in Towards Data Science

Topic Modelling with BERTtopic in Python

Hands-on tutorial on modeling political statements with a state-of-...

Apr 1 290 1



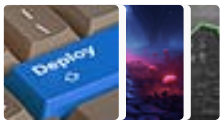
Nakano Kappei

A Novel Approach to Topic Modeling Using Large...

Introduction

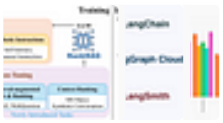
Apr 14 2

Lists



Predictive Modeling w/ Python

20 stories · 1398 saves



Natural Language Processing

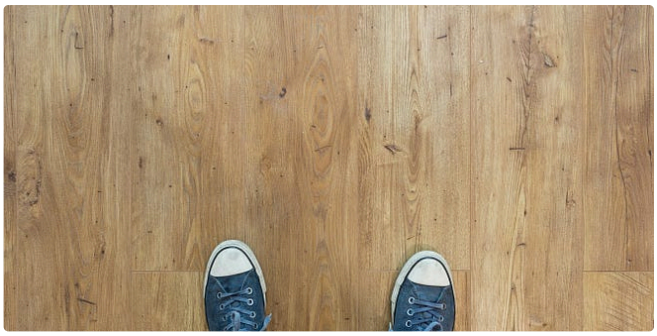
1592 stories · 1148 saves

AI Regulation

6 stories · 516 saves

Practical Guides to Machine Learning

10 stories · 1689 saves

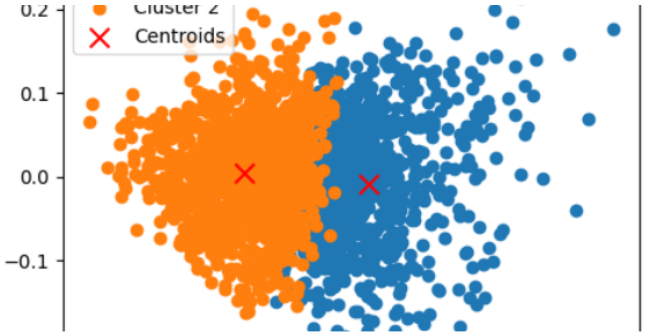


Ednalyn C. De Dios in Data Science Nerd

Topic modeling and network analysis using BERTopic an...

A quick and dirty guide to uncovering themes in domestic violence studies...

Jun 29 80

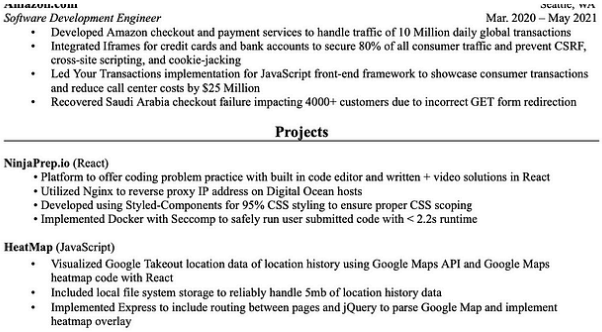


Mohamad Mahmood in Dev Genius

Unsupervised Text Classification Using K-Mea...

Simple, Efficient, Adaptable

Mar 17 14



Software Development Engineer Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by $25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)
- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)
- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay

Alexander Nguyen in Level Up Coding

The resume that got a software engineer a...

1-page. Well-formatted.

Jun 1 14.2K 213

Yu Dong

Topic Summarization and Categorization with GPT

Use GPT-3.5 API for text analytics to categorize and summarize data...

May 25 73

See more recommendations