

Topic Modeling Using Latent Dirichlet Allocation (LDA)

[ALGORITHM](#)[DATA ANALYSIS](#)[NLP](#)[PROJECT](#)[PYTHON](#)[TOPIC MODELING](#)

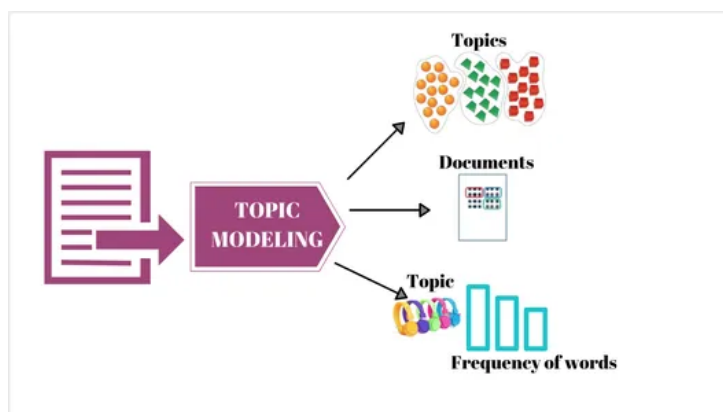
Introduction

The internet is a wealth of knowledge and information, which may confuse readers and make them use more time and energy looking for accurate information about particular areas of interest. To recognize and analyze content in online social networks (OSNs), there is a need for more effective techniques and tools, especially for those who employ user-generated content (UGC) as a source of data.

In NLP(Natural Language Processing), Topic Modeling identifies and extracts abstract topics from large collections of text documents. It uses algorithms such as Latent Dirichlet Allocation (LDA) to identify latent topics in the text and represent documents as a mixture of all the words these topics. Some uses of topic modeling include:

1. Text classification and document organization
2. Marketing and advertising to understand customer preferences
3. Recommendation systems to suggest similar content
4. News categorization and information retrieval systems
5. Customer service and support to categorize customer inquiries.

Latent Dirichlet Allocation, a statistical and visual concept, is used to find the word distribution connections between many documents in a corpus. The Variational Exception Maximization (VEM) technique is used to get the highest probability estimate from the full corpus of text.



Learning Objectives

- This project aims to perform topic modeling on a dataset of news headlines to show the topics that stand out and uncover patterns and trends in the news.
- The second objective of this project will be to have a visual representation of the dominant topics, which news aggregators, journalists, and individuals can use to gain a broad understanding of the current news landscape quickly.

- Understanding the topic modeling pipeline and being able to implement it.

This article was published as a part of the [Data Science Blogathon](#).

Table of Contents

1. [Important Libraries in Topic Modeling Project](#)
2. [Dataset Description of the Topic Modeling Project](#)
3. [Step 1: Importing Necessary Dependencies](#)
4. [Step 2: Importing and Reading Dataset](#)
5. [Step 3: Data Preprocessing](#)
6. [Step 4: Training the model](#)
7. [Step 5: Plotting a Word Cloud for the topics.](#)

Important Libraries in Topic Modeling Project

In a topic modeling project, knowledge of the following libraries plays important roles:

1. **[Gensim](#)**: It is a library for unsupervised topic modeling and document indexing. It provides efficient algorithms for modeling latent topics in large-scale text collections, such as those generated by search engines or online platforms.
2. **[NLTK](#)**: The Natural Language Toolkit (NLTK) is a library for working with human language data. It provides tools for tokenizing, stemming, and lemmatizing text and for performing part-of-speech tagging, named entity recognition, and sentiment analysis.
3. **[Matplotlib](#)**: It is a plotting library for Python. It is used for visualizing the results of topic models, such as the distribution of topics over documents or the relationships between words and topics.
4. **[Scikit-learn](#)**: It is a library for machine learning in Python. It provides a wide range of algorithms for modeling topics, including Latent Dirichlet Allocation (LDA), Non-Negative Matrix Factorization (NMF), and others.
5. **[Pandas](#)**: It is a library for data analysis in Python. It provides data structures and functions for working with structured data, such as the results of topic models, in a convenient and efficient manner.



What is Topic Modeling Used For?

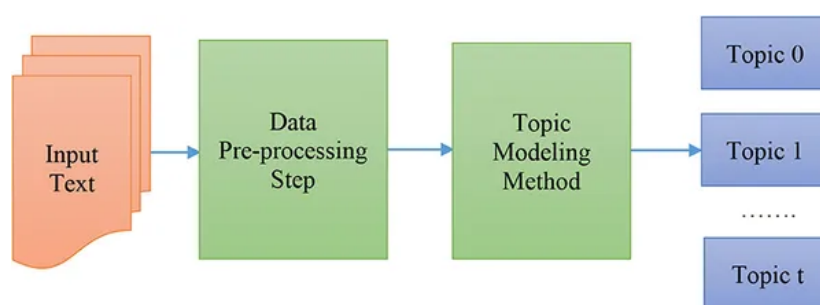
Topic modeling is a versatile [technique](#) utilized in natural language processing and machine learning to uncover underlying themes or topics within a corpus of documents. It serves various purposes seamlessly:

1. **Document Organization:** Topic modeling aids in organizing extensive document collections by naturally grouping them into clusters based on prevalent themes, facilitating efficient management and retrieval.
2. **Information Retrieval:** Enhancing search engines, topic modeling categorizes documents into topics, enabling users to access relevant information swiftly and accurately.
3. **Content Recommendation:** Online platforms leverage topic modeling to recommend personalized content to users, enhancing user experience and engagement across diverse domains such as news, e-commerce, and social media.
4. **Text Summarization:** By discerning primary topics within documents, topic modeling streamlines the process of generating concise summaries that encapsulate the core essence of the text, fostering comprehension and accessibility.
5. **Understanding Textual Data:** Researchers and analysts employ topic modeling to glean insights from vast troves of textual data, uncovering prevalent themes, trends, and patterns, thereby enriching comprehension and decision-making processes.
6. **Sentiment Analysis:** Topic modeling, when integrated with sentiment analysis techniques, enables the exploration of sentiment nuances associated with different topics, facilitating a deeper understanding of textual data and its emotional context.

In essence, topic modeling serves as a powerful tool in deciphering the intricate fabric of textual data, offering invaluable insights and facilitating a myriad of applications across artificial intelligence, data analysis, and information retrieval domains.

Dataset Description of the Topic Modeling Project

The dataset used is from Kaggle's [A million News Headlines](#). The data contains 1.2 million rows and 2 columns namely "publish date" and "headline text". "Headline text" column contains news headlines and "publish date" column contains the date the headline was published.



Step 1: Importing Necessary Dependencies

The code below imports the libraries(listed in the introduction section above) needed for our project.

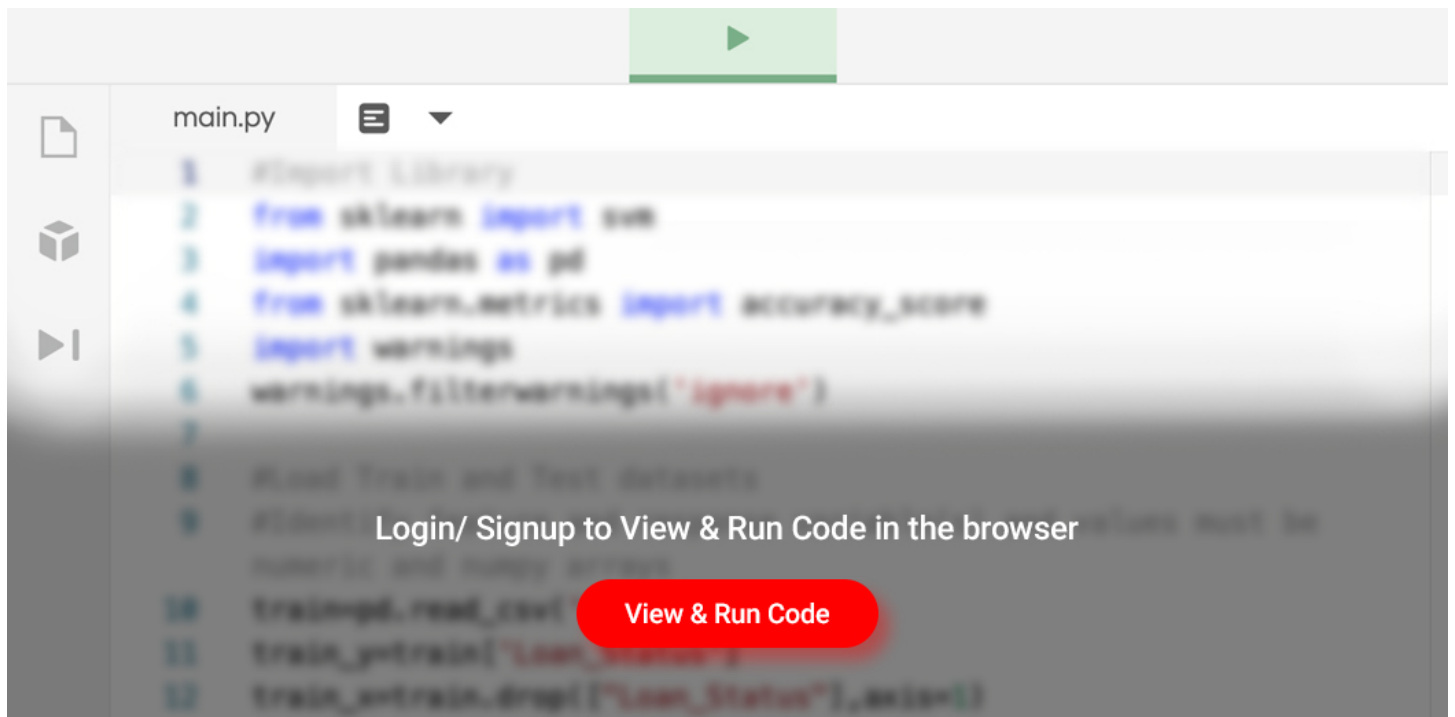
```
import pandas as pd import matplotlib.pyplot as plt import nltk from nltk.tokenize import word_tokenize from nltk.stem import WordNetLemmatizer from nltk.corpus import stopwords import gensim from gensim.corpora import Dictionary from gensim.models import LdaModel from gensim.matutils import corpus2csc from sklearn.feature_extraction.text import CountVectorizer from wordcloud import WordCloud import matplotlib.pyplot as plt
```

Step 2: Importing and Reading Dataset

Loading our dataset that is in csv format into a [data frame](#). The code below loads the 'abcnews-date-text.csv' file into a data frame named 'df'.

```
#loading the file from its local path into a dataframe df=pd.read_csv(r"path\abcnews-date-text.csv\abcnews-date-text.csv") df
```

Python Code:



Output:

Out[71]:	publish_date	headline_text
0	20030219	aba decides against community broadcasting lic...
1	20030219	act fire witnesses must be aware of defamation
2	20030219	a g calls for infrastructure protection summit
3	20030219	air nz staff in aust strike for pay rise
4	20030219	air nz strike to affect australian travellers
...
1244179	20211231	two aged care residents die as state records 2...
1244180	20211231	victoria records 5,919 new cases and seven deaths
1244181	20211231	wa delays adopting new close contact definition
1244182	20211231	western ringtail possums found badly dehydrate...
1244183	20211231	what makes you a close covid contact here are ...
1244184 rows x 2 columns		

Step 3: Data Preprocessing

The code below selects the first 1,000,000 rows in the dataset and drops the rest of the columns except the "headline text" column and then names the new data-frame 'data.'

```
data = df.sample(n=100000, axis=0) #to select only a million rows to use in our dataset
data = data['headline_text'] #to extract the headline_text column and give it the variable name data
```

Next, we perform lemmatization and removal of stop-words from the data.

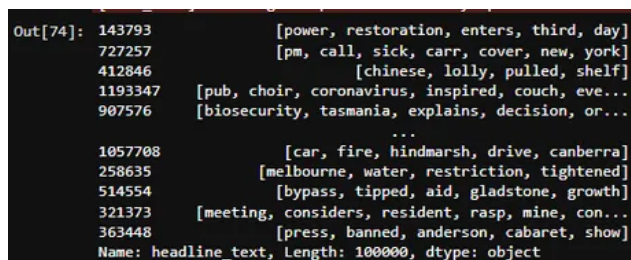
Lemmatization reduces words to the base root, reducing the [dimensionality and complexity](#) of the textual data. We assign WordNetLemmatizer() to the variable. This is important to improve the algorithm's performance and helps the algorithm focus on the meaning of the words rather than the surface form.

Stop-words are common words like “the” and “a” that often appear in text data but do not carry lots of meaning. Removing them helps reduce the data's complexity, speeds up the algorithm, and makes it easier to find meaningful patterns.

The code below downloads dependencies for performing lemmatization and removing stop-words, then defines a function to process the data and finally applies the function to our data-frame 'data.'

```
# lemmatization and removing stopwords #downloading dependencies nltk.download('punkt')
nltk.download('wordnet') nltk.download('stopwords') lemmatizer = WordNetLemmatizer() stop_words =
set(stopwords.words("english")) #function to lemmatize and remove stopwords from the text data def
preprocess(text): text = text.lower() words = word_tokenize(text) words = [lemmatizer.lemmatize(word) for
word in words if word not in stop_words] return words #applying the function to the dataset data =
data.apply(preprocess) data
```

Output:



```
Out[74]: 143793      [power, restoration, enters, third, day]
          727257      [pm, call, sick, carr, cover, new, york]
          412846      [chinese, lolly, pulled, shelf]
          1193347     [pub, choir, coronavirus, inspired, couch, eve...
          907576     [biosecurity, tasmania, explains, decision, or...
          ...
          1057708     [car, fire, hindmarsh, drive, canberra]
          258635      [melbourne, water, restriction, tightened]
          514554      [bypass, tipped, aid, gladstone, growth]
          321373      [meeting, considers, resident, rasp, mine, con...
          363448      [press, banned, anderson, cabaret, show]
          Name: headline_text, Length: 100000, dtype: object
```

Step 4: Training the Model

The number of [topics](#) is set to 5 (which can be set to as many topics as one wants to extract from the data), the number of passes is 20, and the alpha and eta are set to “auto.” This lets the model estimate the appropriate values. You can experiment with different parameters to see the impact on results.

The code below processes the data to remove words that appear in fewer than 5 documents and those that appear in more than 50% of the data. This ensures that the model does not include words that appear less in the data or more in the data. For example, news headlines in a country will have a lot of mentions of that country which will alter the effectiveness of our model. Then we create a corpus from the filtered data. We then select the number of topics and train the Lda-model, get the topics from the model using ‘show topics’, and then print the topics.

```
# Create a dictionary from the preprocessed data dictionary = Dictionary(data) # Filter out words that appear
in fewer than 5 documents or more than 50% of the documents dictionary.filter_extremes(no_below=5,
no_above=0.5) bow_corpus = [dictionary.doc2bow(text) for text in data] # Train the LDA model num_topics = 5
ldamodel = LdaModel(bow_corpus, num_topics=num_topics, id2word=dictionary, passes=20, alpha='auto',
eta='auto') # Get the topics topics = ldamodel.show_topics(num_topics=num_topics, num_words=10, log=False,
```

```
formatted=False) # Print the topics for topic_id, topic in topics: print("Topic: {}".format(topic_id))
print("Words: {}".format([word for word, _ in topic]))
```

Output:

```
Topic: 0
Words: ['council', 'win', 'interview', 'home', 'sa', 'world', 'claim', 'first', 'test', 'talk']
Topic: 1
Words: ['police', 'man', 'court', 'woman', 'death', 'crash', 'car', 'hit', 'face', 'child']
Topic: 2
Words: ['new', 'say', 'u', 'call', 'australia', 'plan', 'fire', 'back', 'govt', 'water']
Topic: 3
Words: ['nsw', 'year', 'day', 'take', 'election', 'set', 'one', 'road', 'record', 'final']
Topic: 4
Words: ['australian', 'sydney', 'get', 'attack', 'may', 'market', 'farmer', 'killed', 'open', 'rise']
```

Step 5: Plotting a Word Cloud for the Topics

Word cloud is a [data visualization tool](#) used to visualize the most frequently occurring words in a large amount of text data and can be useful in understanding the topics present in data. It’s important in text data analysis, and it provides valuable insights into the structure and content of the data.

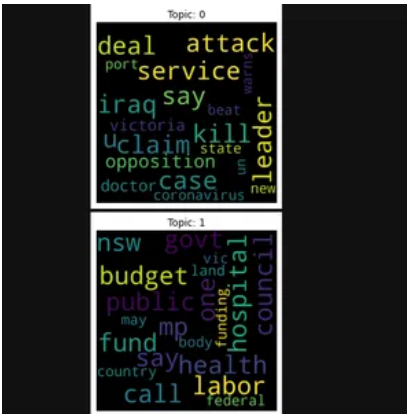
Word cloud is a simple but effective way of visualizing the content of large amounts of text data. It displays the most frequent words in a graphical format, allowing the user to easily identify the key topics and themes present in the data. The size of each word in the word cloud represents its frequency of occurrence so that the largest words in the cloud correspond to the most commonly occurring words in the data.

This visualization tool can be a valuable asset in text data analysis, providing an easy-to-understand representation of the data’s content. For example, a word cloud can be used to quickly identify the dominant topics in a large corpus of news articles, customer reviews, or social media posts. This information can then guide further analysis, such as sentiment analysis or topic modeling, or inform decision-making, such as product development or marketing strategy.

The code below plots word clouds using topic words from the topic id using matplotlib.

```
# Plotting a wordcloud of the topics for topic_id, topic in
enumerate(lda_model.print_topics(num_topics=num_topics, num_words=20)):
    topic_words = "
    ".join([word.split(" ")[1].strip() for word in topic[1].split(" + ")])
    wordcloud = WordCloud(width=800, height=800, random_state=21, max_font_size=110).generate(topic_words)
    plt.figure()
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.title("Topic: {}".format(topic_id))
    plt.show()
```

Output:





How will LDA optimize the distributions?

Latent Dirichlet Allocation (LDA) is a generative probabilistic [model used for topic modeling](#), which is the process of identifying topics present in a collection of documents. LDA optimizes the distributions through an iterative process that involves estimating the parameters of the model based on the observed data (i.e., the words in the documents). In LDA, each document is viewed as a mixture of topics, and each topic is characterized by a distribution of words. This is represented mathematically through the concept of “topic word distribution.” Essentially, LDA assigns a probability distribution over words for each topic, and for each document, there is a “topic distribution” indicating the probability of each topic being present in that document. By analyzing these distributions, LDA determines the best member-only stories, where “best” refers to the most likely topics associated with a given document. The process involves calculating the posterior distribution of topics given the observed words in the document, and the equation governing this process encapsulates the essence of LDA.

let's simplify it:

1. **Getting Started:** LDA starts by guessing what topics might be in the documents and what words might belong to those topics.
2. **Guess and Check:** It then looks at each word in each document and makes a guess about which topic it might belong to.
3. **Adjusting:** Based on those guesses, LDA adjusts its ideas about which topics are in the documents and which words belong to each topic.
4. **Repeating:** It keeps doing this, guessing and adjusting, over and over again, until it's not changing its ideas much anymore.
5. **Figuring Out:** Once it's done adjusting, LDA figures out the final topics and which words belong to each topic based on its best guesses

Conclusion

Topic modeling is a [powerful tool for analyzing](#) and understanding large collections of text data. Topic modeling works by discovering latent topics and the relationships between words and documents, can help uncover hidden patterns and trends in text data and provide valuable insights into the underlying structure of text data.

The combination of [powerful libraries](#) such as Gensim, NLTK, Matplotlib, scikit-learn, and Pandas make it easier to perform topic modeling and gain insights from text data. As the amount of text data generated by individuals, organizations, and society continues to grow, the importance of topic modeling and its role in data analysis and understanding is only set to increase.

Feel free to leave your comments, and I hope the article has provided insights into topic modeling with Latent Dirichlet Allocation (LDA) and the various use cases of this algorithm.

The code can be found in my [github repository](#).

The media shown in this article is not owned by Analytics Vidhya and is used at the Author's discretion.

Article Url - <https://www.analyticsvidhya.com/blog/2023/02/topic-modeling-using-latent-dirichlet-allocation-lda/>



[Kevin Kibe](#)