Topic Modeling

# Definition

Topic modeling is a form of text mining that employs unsupervised and supervised statistical machine learning techniques to identify patterns in a corpus or large amount of unstructured text. It can take your huge collection of documents and group the words into clusters of words, identify topics, by using a process of similarity.

In other words, topic modeling is an unsupervised machine learning technique that’s capable of scanning a set of documents, detecting word and phrase patterns within them, and automatically clustering word groups and similar expressions that best characterize a set of documents.

Topic modeling identifies repeated phrases and contextual cues to find common themes or categories of content in a collection of long-form or short-form content. You can use the model to find topics and related terms covered in a set of documents or group texts by their themes.

# Models and algorithms

There are different types of topic modeling models. According to their underlying modeling techniques, two categories of topic models are recognized: probabilistic, and non-probabilistic.

## Probabilistic topic modeling

Probabilistic topic modeling is a statistical technique that is used to identify topics in a collection of documents. It is based on the assumption that each document contains a mixture of topics and that each word in the document is generated by one of the topics. The goal of probabilistic topic modeling is to estimate the topic mixture for each document and the word distribution for each topic.

Latent Dirichlet Allocation (LDA) is one of the most popular probabilistic topic modeling algorithms.

## Non-probabilistic topic modeling

Non-probabilistic topic modeling methods are based on matrix factorization techniques such as Non-negative Matrix Factorization (NMF) and Singular Value Decomposition (SVD). These methods do not assume any probabilistic generative model for the data.

Some of the most common techniques of topic modeling include Latent Dirichlet Allocation (LDA), Latent Semantic Analysis (LSA), Non-Negative Matrix Factorization (NMF), Parallel Latent Dirichlet Allocation (PLDA), Pachinko Allocation Model and Singular Value Decomposition (SVD).

# Implementation

## Data

A collection of 50,000 written articles have been collected into a .CSV file from publications like: the New York Times, Breitbart, CNN, Business Insider, the Atlantic, and Fox News.

The data primarily falls between the years of 2016 and July 2017, although there is a not-insignificant number of articles from 2015, and a possibly insignificant number from before then.

Each article has its own data including:

* Author name
* Publication
* Date of publication
* Title
* Content

## Libraries implemented

* 1. Pandas

Pandas is a popular Python library for data manipulation and analysis. It provides data structures like Series and Datagram, which offer powerful indexing and reshaping capabilities. Pandas supports data cleaning, preprocessing, and transformation tasks. It can read and write data in various file formats, handles time series data, and integrates well with other Python libraries. Pandas is widely used in data analysis, data science, and machine learning.

* 1. NLTK

NLTK is a popular Python library for natural language processing (NLP). It offers tools for text preprocessing, part-of-speech tagging, named entity recognition, sentiment analysis, language modeling, and text classification. NLTK provides linguistic resources and integrates with other machine learning libraries. It is widely used in research, education, and industry for NLP tasks.

* 1. Gensim

Gensim is a popular Python library for topic modeling and NLP tasks. It supports LDA and other topic models, document similarity, and indexing. Gensim offers text preprocessing, word vector representations, and document modeling. It is memory-efficient, integrates with other libraries, and is widely used for large-scale text analysis in research and industry.

* 1. Seaborn

Seaborn is a Python data visualization library known for its attractive default themes and simplified customization options. It specializes in statistical plots, making it easy to explore relationships and distributions in data. Seaborn integrates well with Pandas and offers support for categorical data and advanced visualization techniques. It is widely used for creating visually appealing and informative plots in data analysis and storytelling.

* 1. pyLDAvis

PyLDAvis is a Python library that provides interactive visualizations for exploring and interpreting topic models, particularly LDA models. It offers tools for topic interpretation, coherence assessment, and integrates with popular topic modeling libraries. PyLDAvis allows users to interactively explore topics, keywords, and topic relationships, making it a valuable tool for gaining insights from topic modeling results.

* 1. Pickle

Pickle is a Python library used for object serialization and deserialization. It allows Python objects to be converted into a byte stream for storage or transmission, and later reconstructed. Pickle provides persistence, interoperability, and flexibility, and integrates well with other libraries.

## Preprocessing

1. Remove duplicate entries (around 2,000 articles removed).
2. Remove all metadata leaving only the article content to reduce clutter.
3. Create a dictionary of stop words.
4. Convert all text to lowercase.
5. Tokenize each article string into separate tokens for further processing.
6. Lemmatize each token.
7. Check if the token is in the stop words dictionary.
8. Remove tokens that are purely: numbers, symbols or punctuation.
9. Remove any empty entries after preprocessing.
10. Create a dictionary and a corpus with the processed tokens using bag of words method.

## Determining number of topics

To determine the optimal number of topics in an LDA (Latent Dirichlet Allocation) model using coherence scores, you can follow these steps:

1. Preprocess your text data: Clean and preprocess your text data by removing stop words, punctuation, and performing stemming or lemmatization if necessary. This step helps to improve the quality of the topics extracted by the LDA model.
2. Build an LDA model: Train an LDA model on your preprocessed text data. Specify a range of candidate topic numbers to explore during model selection. The number of topics is typically an input parameter in LDA models.
3. Compute coherence scores: For each candidate number of topics, calculate the coherence score. Coherence scores measure the interpretability and semantic coherence of topics. The higher the coherence score, the better the topics are considered.
4. Evaluate coherence scores: Plot the coherence scores for different numbers of topics. This can be done using a line plot or a bar plot, with the number of topics on the x-axis and the coherence score on the y-axis. Look for the "elbow" point in the plot where the coherence score starts to plateau.
5. Choose the optimal number of topics: Select the number of topics at the point where the coherence score plateaus or reaches a peak. This indicates the point where adding more topics does not significantly improve the coherence of the model.

## The LDA model

The LDA model was chosen for a number of reasons:

* 1. Unsupervised learning: LDA is an unsupervised learning algorithm, meaning it does not require labeled training data. It can discover topics solely based on the patterns and co-occurrence of words in the documents, making it applicable to a wide range of text analysis tasks.
  2. Flexibility: LDA allows documents to have multiple topics, meaning that a document can discuss multiple themes simultaneously. This flexibility makes it suitable for analyzing complex and diverse text data where documents may cover multiple subject areas.
  3. Scalability: LDA scales well to large text collections. Efficient algorithms and implementations, such as distributed versions, allow processing of massive datasets with millions of documents and thousands of topics.
  4. Scalability: LDA scales well to large text collections. Efficient algorithms and implementations, such as distributed versions, allow processing of massive datasets with millions of documents and thousands of topics.

It is worth noting that while LDA has many advantages, it also has limitations. For example, it assumes a bag-of-words representation, disregarding word order and syntax. Additionally, LDA may struggle with short documents or noisy data. Therefore, it is important to consider these factors and tailor the use of LDA to the specific characteristics of your text data and application requirements.

To train an LDA (Latent Dirichlet Allocation) model, you can follow these steps:

* 1. Prepare your text data: Ensure that your text data is in a format suitable for training the LDA model. Clean the text by removing any irrelevant information, such as HTML tags or special characters. Tokenize the text into individual words or terms.
  2. Create a document-term matrix: Convert your preprocessed text data into a document-term matrix. Each row represents a document, and each column represents a term from the vocabulary. The values in the matrix typically represent term frequencies or term weights (e.g., TF-IDF scores).
  3. Choose the number of topics: Determine the desired number of topics for your LDA model. This can be based on prior knowledge or using techniques like coherence scores (as mentioned in the previous response).
  4. Train the LDA model: Use a suitable library or implementation to train the LDA model. Popular libraries for LDA include gensim (Python) and the topicmodels package in R. Typically, you will need to provide the document-term matrix, the number of topics, and other parameters such as the number of iterations and the random seed.
  5. Explore the trained model: Once the LDA model is trained, you can explore the results. This includes accessing the learned topic-word distributions and document-topic distributions. These distributions provide insights into the most probable words for each topic and the topic composition of each document.
  6. Interpret and evaluate the topics: Analyze the topics generated by the LDA model and assign human-readable labels based on the most probable words. Evaluate the topics for coherence, relevance, and interpretability. You can also use various evaluation metrics, such as perplexity or topic coherence, to assess the quality of the model.
  7. Apply the LDA model: Once you are satisfied with the trained LDA model, you can apply it to new documents or use it for topic inference. This allows you to assign topics to unseen documents based on the learned topic distributions.