Medium Articles Clustering

Problem Statement

Background: Medium is an online publishing platform that allows users to create and share articles on various topics. With millions of articles published on Medium, it can be challenging to navigate and explore the content efficiently. Clustering articles based on their topics can provide a useful organization and recommendation system for users.

Dataset: The dataset used for this project consists of Medium articles. Each article is represented by its text, author, and number of claps (a measure of popularity). The dataset contains information about various authors and their articles, covering a wide range of topics.

Objective: The objective of this project is to cluster the Medium articles based on their topics using the Latent Dirichlet Allocation (LDA) algorithm. LDA is a popular topic modeling technique that discovers hidden topics in a collection of documents based on the distribution of words. By clustering the articles, we can gain insights into the prevalent topics on Medium and provide users with a more organized and personalized reading experience.

Function: top_n_topics The top_n_topics function takes the following parameters:

- n: An integer representing the number of topics to extract and display.
- articles_df: A pandas DataFrame containing the Medium articles data. It should have columns for "text", "author", and "claps".

The function performs the following steps:

Preprocess the articles:

- Convert the text to lowercase.
- Expand contractions (e.g., "don't" to "do not").
- Remove stopwords and punctuation.
- Lemmatize the words.
- Remove any additional marks or symbols.
- Tokenize the preprocessed articles.
- Create a dictionary and corpus for the tokenized articles using the Gensim library.
- Build an LDA model with the specified number of topics using the Gensim library.
- Calculate the coherence score of the LDA model to evaluate the quality of the topics.
- Visualize the topics using the pyLDAvis library.
- Return the top n topics along with their keywords and coherence scores.

The top_n_topics function provides a convenient way to extract and explore the main topics from the Medium articles dataset. It helps users gain insights into the prevalent themes and facilitates content recommendation based on their interests.

Framework

Outline: Clustering Medium Articles

Import Libraries

- Import the necessary libraries, including pandas, nltk, gensim, pyLDAvis, and matplotlib.

Load and Preprocess the Dataset

- Load the Medium articles dataset into a pandas DataFrame.
- Perform data cleaning and preprocessing on the text column, including converting to lowercase, expanding contractions, removing stopwords and punctuation, lemmatizing words, and removing any additional marks or symbols.

Tokenize the Preprocessed Articles

- Tokenize the preprocessed articles using the nltk library.
- Create a new column in the DataFrame to store the tokenized text.

Create Dictionary and Corpus

- Create a dictionary from the tokenized articles using the gensim library.
- Create a corpus by converting the tokenized articles into bag-of-words format using the dictionary.

Build and Evaluate LDA Model

- Build an LDA model using the gensim library, providing the corpus, dictionary, and the desired number of topics as input.
- Calculate the coherence score of the LDA model using the gensim library to evaluate the quality of the topics.

Visualize the Topics

- Visualize the topics using the pyLDAvis library to gain insights and interpretability.
- Display the interactive visualization using matplotlib.

Define the top_n_topics Function

- Define the top_n_topics function that takes parameters for the number of topics to extract and the DataFrame containing the Medium articles data.
- Implement the preprocessing, tokenization, dictionary, corpus, LDA model building, coherence score calculation, and topic visualization steps within the function.
- Return the top n topics along with their keywords and coherence scores.

Test the top_n_topics Function

- Use the top_n_topics function to extract and display the top topics from the Medium articles dataset.
- Pass the desired number of topics and the DataFrame containing the articles data as parameters to the function.
- Print the results to observe the top topics, their associated keywords, and coherence scores.

Main Execution

- Call the main execution function that loads the dataset, preprocesses the articles, and calls the top_n_topics function to extract and display the top topics.
- Pass the appropriate parameters to the top_n_topics function and print the results.

By following this outline, one can systematically write the code to cluster Medium articles based on topics using the Latent Dirichlet Allocation (LDA) algorithm and extract the top topics from the dataset.

Code Explanation

The code you provided aims to cluster Medium articles based on topics using the Latent Dirichlet Allocation (LDA) algorithm. Let's break down the code and understand its workflow:

Importing Libraries: The code starts by importing the necessary libraries. These libraries provide various functionalities required for data preprocessing, topic modeling, and visualization. For example, the pandas library is used for data manipulation, nltk for natural language processing tasks, gensim for topic modeling, pyLDAvis for topic visualization, and matplotlib for creating plots.

Loading and Preprocessing the Dataset: The code loads the Medium articles dataset into a pandas DataFrame. This dataset contains articles as text. The text is then preprocessed to make it suitable for topic modeling. Preprocessing involves converting text to lowercase, expanding contractions, removing stopwords (common words like "and," "the," etc.), removing punctuation marks, lemmatizing words (converting words to their base form), and removing any additional marks or symbols.

Tokenizing the Preprocessed Articles: In this step, the preprocessed articles are tokenized using the nltk library. Tokenization means splitting the text into individual words or tokens. The tokens are then stored in a new column in the DataFrame.

Creating Dictionary and Corpus: The code creates a dictionary from the tokenized articles using the gensim library. A dictionary maps each unique word in the articles to a unique numerical ID. Additionally, the code converts the tokenized articles into a bag-of-words format, which represents each article as a list of word-frequency pairs. This collection of word-frequency pairs is called the corpus.

Building and Evaluating LDA Model: The code builds an LDA model using the gensim library. LDA is a popular topic modeling algorithm that discovers latent topics in a collection of documents. The LDA model is trained using the corpus and the dictionary created in the previous steps. The code also calculates the coherence score of the LDA model using the gensim library. The coherence score measures the interpretability and quality of the topics generated by the model.

Visualizing the Topics: Next, the code visualizes the topics using the pyLDAvis library. This library provides an interactive visualization that helps understand and interpret the topics. The visualization includes a bar chart representing the prevalence of each topic and an intertopic distance map that shows the similarity between topics. The visualization is displayed using the matplotlib library.

Defining the top_n_topics Function: The code defines a function called top_n_topics that takes parameters for the number of topics to extract and the DataFrame containing the Medium articles data. This function encapsulates the preprocessing, tokenization, dictionary creation, corpus creation, LDA model building, coherence score calculation, and topic visualization steps.

Testing the top_n_topics Function: The code tests the top_n_topics function by calling it with the desired number of topics and the DataFrame containing the articles data. The function extracts the top topics along with their keywords and coherence scores. The results are then printed, allowing us to observe the top topics and their associated keywords.

Main Execution: The code calls the main execution function, which loads the dataset, preprocesses the articles, and calls the top_n_topics function to extract and display the top topics. The function parameters are passed appropriately, and the results are printed.

By following this code, one can cluster Medium articles based on topics using LDA and extract the top topics from the dataset. It's an exciting way to explore and analyze textual data, gaining insights into the prevalent themes and subjects covered in the articles.

Future Work

The code you provided aims to cluster Medium articles based on topics using the Latent Dirichlet Allocation (LDA) algorithm. Let's break down the code and understand its workflow:

Importing Libraries: The code starts by importing the necessary libraries. These libraries provide various functionalities required for data preprocessing, topic modeling, and visualization. For example, the pandas library is used for data manipulation, nltk for natural language processing tasks, gensim for topic modeling, pyLDAvis for topic visualization, and matplotlib for creating plots.

Loading and Preprocessing the Dataset: The code loads the Medium articles dataset into a pandas DataFrame. This dataset contains articles as text. The text is then preprocessed to make it suitable for topic modeling. Preprocessing involves converting text to lowercase, expanding contractions, removing stopwords (common words like "and," "the," etc.), removing punctuation marks, lemmatizing words (converting words to their base form), and removing any additional marks or symbols.

Tokenizing the Preprocessed Articles: In this step, the preprocessed articles are tokenized using the nltk library. Tokenization means splitting the text into individual words or tokens. The tokens are then stored in a new column in the DataFrame.

Creating Dictionary and Corpus: The code creates a dictionary from the tokenized articles using the gensim library. A dictionary maps each unique word in the articles to a unique numerical ID. Additionally, the code converts the tokenized articles into a bag-of-words format, which represents each article as a list of word-frequency pairs. This collection of word-frequency pairs is called the corpus.

Building and Evaluating LDA Model: The code builds an LDA model using the gensim library. LDA is a popular topic modeling algorithm that discovers latent topics in a collection of documents. The LDA model is trained using the corpus and the dictionary created in the previous steps. The code also calculates the coherence score of the LDA model using the gensim library. The coherence score measures the interpretability and quality of the topics generated by the model.

Visualizing the Topics: Next, the code visualizes the topics using the pyLDAvis library. This library provides an interactive visualization that helps understand and interpret the topics. The visualization includes a bar chart representing the prevalence of each topic and an intertopic distance map that shows the similarity between topics. The visualization is displayed using the matplotlib library.

Defining the top_n_topics Function: The code defines a function called top_n_topics that takes parameters for the number of topics to extract and the DataFrame containing the Medium articles data. This function encapsulates the preprocessing, tokenization, dictionary creation, corpus creation, LDA model building, coherence score calculation, and topic visualization steps.

Testing the top_n_topics Function: The code tests the top_n_topics function by calling it with the desired number of topics and the DataFrame containing the articles data. The function extracts the top topics along with their keywords and coherence scores. The results are then printed, allowing us to observe the top topics and their associated keywords.

Main Execution: The code calls the main execution function, which loads the dataset, preprocesses the articles, and calls the top_n_topics function to extract and display the top topics. The function parameters are passed appropriately, and the results are printed.

By following this code, one can cluster Medium articles based on topics using LDA and extract the top topics from the dataset. It's an exciting way to explore and analyze textual data, gaining insights into the prevalent themes and subjects covered in the articles.

Concept Explanation

Imagine you're at a buffet with a variety of dishes. You're not sure what the ingredients are, but you want to figure out the main themes or topics that these dishes represent. You don't want to taste each dish individually because that would take forever, and you might end up with a tummy ache!

That's where Latent Dirichlet Allocation (LDA) comes to the rescue. LDA is like a smart food critic that analyzes a bunch of dishes and tells you the underlying themes without having to taste them all. How cool is that?

Here's how it works:

The Ingredients of LDA: LDA assumes that each dish (document) in the buffet is made up of a mixture of different ingredients (topics). These ingredients are represented by words. For example, a dish about sports might have ingredients like "football," "basketball," and "soccer."

Mixing the Ingredients: LDA also assumes that each dish has a specific proportion of ingredients. Some dishes might be more focused on sports, while others might have a mix of sports, technology, and cats (yes, cats can be an ingredient too!). This mixing of ingredients creates a unique recipe for each dish.

The LDA Chef at Work: Now, imagine an LDA chef in the kitchen. This chef's goal is to reverse-engineer the dishes and figure out the ingredients and their proportions. The chef does this by observing the words in the dishes and trying to untangle the underlying themes.

The LDA Recipe: The LDA chef has a recipe to follow. First, the chef randomly assigns ingredients to each dish. Then, the chef looks at the words in the dishes and makes adjustments to the ingredient assignments. For example, if the chef sees words like "touchdown," "goal," and "slam dunk," they'll think, "Aha! This dish is all about sports!" The chef repeats this process multiple times to fine-tune the ingredient assignments and proportions.

Uncovering the Topics: Once the LDA chef finishes cooking, they reveal the topics they discovered. Each topic is a combination of ingredients (words) and their proportions. For example, a topic about sports might have "football" with a 50% proportion, "basketball" with a 30% proportion, and "soccer" with a 20% proportion.

Topic Sorting: Finally, the chef sorts the topics based on their proportions. The topics with higher proportions are considered more prominent in the buffet. So, if the chef found that the topic about sports has a higher proportion compared to the topic about technology, they'll say, "Hey, this buffet is more focused on sports!"

Now, isn't that a tasty way to discover the hidden themes in a pile of documents?

In our project, we used LDA to analyze a collection of Medium articles. Instead of dishes, we had articles, and instead of ingredients, we had topics represented by words. LDA helped us uncover the main topics and their proportions in the articles, giving us a deeper understanding of the dataset.

Remember, LDA is like a food critic that helps us make sense of the buffet without having to taste every dish. So, grab your metaphorical chef hat and let LDA cook up some delicious insights for you!

Exercise Questions

Exercise 1: Topic Distribution

Question: Can you explain how to interpret the topic distribution obtained from the LDA analysis?

Answer: The topic distribution represents the proportions of each topic in a given document. It tells us how much of each topic is present in the document. For example, if we have three topics: Sports, Technology, and Food, and we obtain a topic distribution of [0.3, 0.5, 0.2] for a document, it means that the document contains approximately 30% Sports, 50% Technology, and 20% Food. The sum of the proportions should be close to 1, as it represents the entire document.

Exercise 2: Finding Dominant Topics

Question: How can we determine the dominant topic for a document based on the topic distribution?

Answer: To find the dominant topic for a document, we look for the topic with the highest proportion in the topic distribution. We identify the topic that has the largest value among all the topics in the distribution. For example, if the topic distribution is [0.2, 0.5, 0.3], the dominant topic would be the one with the proportion of 0.5, which indicates it is the most significant topic in that document.

Exercise 3: Optimal Number of Topics

Question: How can we determine the optimal number of topics to use in an LDA analysis?

Answer: Determining the optimal number of topics is crucial for obtaining meaningful results. One common approach is to use techniques such as perplexity or coherence scores. Perplexity measures how well the model predicts the held-out data. Lower perplexity scores indicate better performance. Coherence scores, on the other hand, evaluate the interpretability and semantic coherence of the topics. Higher coherence scores suggest more meaningful topics. By experimenting with different numbers of topics and analyzing the corresponding perplexity and coherence scores, we can identify the optimal number of topics that balance interpretability and performance.

Exercise 4: Topic Coherence

Question: What is topic coherence, and why is it important in LDA?

Answer: Topic coherence measures the interpretability and coherence of the topics generated by the LDA model. It evaluates how well the words within each topic are related to each other. Higher topic coherence indicates that the topics are more meaningful and coherent. It is essential in LDA because we want the topics to be easily interpretable and representative of distinct themes. Without high coherence, the topics may be vague or overlapping, making it difficult to extract meaningful insights from the analysis.

Exercise 5: Limitations of LDA

Question: What are some limitations of the LDA algorithm?

Answer: While LDA is a powerful tool for topic modeling, it has some limitations. Here are a few:

- Bag-of-Words Assumption: LDA treats each document as a bag of words, disregarding the order or context of the words. This can lead to a loss of important semantic information.
- Fixed Number of Topics: LDA requires specifying the number of topics in advance, which can be challenging. Choosing an inappropriate number may result in topics that are either too specific or too broad.
- Sensitivity to Preprocessing: LDA is sensitive to the quality of preprocessing steps, such as tokenization, stop-word removal, and stemming. Poor preprocessing can affect the quality of the generated topics.
- Difficulty Handling Short Documents: LDA may struggle with short documents that lack sufficient context to capture meaningful topics accurately.
- Topic Overlapping: Topics generated by LDA can overlap, making it challenging to assign exclusive themes to certain topics.

Understanding these limitations can help us make informed decisions and address potential challenges when using LDA for topic modeling.