Stories Clustering

Problem Statement

Problem Description:

In the Stories Clustering project, the goal is to analyze a collection of textual stories and cluster them based on their underlying topics. The project aims to uncover the latent themes present in the stories and group them together to provide insights and facilitate further analysis. By clustering the stories, we can identify patterns, similarities, and differences among different narratives, enabling us to gain a deeper understanding of the dataset.

Background Information:

The dataset used in this project consists of a collection of textual stories from various sources. These stories may cover a wide range of topics, such as news articles, blog posts, short stories, or user-generated content. Each story is represented as a text document, which may vary in length and complexity.

Topic modeling is a popular technique in natural language processing that allows us to extract the underlying themes or topics from a collection of documents. It is particularly useful when dealing with large text corpora, where manual analysis becomes impractical. By applying topic modeling algorithms, we can automatically identify the main topics within the stories and group them into meaningful clusters.

In this project, we utilize several topic modeling algorithms, including Latent Semantic Indexing (LSI), Hierarchical Dirichlet Process (HDP), and Latent Dirichlet Allocation (LDA). These algorithms use different approaches to uncover the latent topics in the dataset. LSI focuses on identifying semantic relationships between words and documents, HDP

determines the number of topics automatically, and LDA probabilistically assigns topics to documents based on statistical patterns.

Additionally, we employ preprocessing techniques to clean the text data before applying the topic modeling algorithms. This includes removing stopwords, punctuation, and numbers, as well as lemmatizing words to their base form. By preprocessing the data, we aim to reduce noise and improve the quality of the resulting topic models.

To visualize and interpret the topics generated by the algorithms, we utilize the pyLDAvis library. pyLDAvis provides an interactive web-based visualization that helps us explore the relationships between topics, identify key terms for each topic, and gain insights into the overall structure of the topic models. This visualization aids in understanding the underlying themes present in the stories and facilitates further analysis and interpretation.

Dataset Information:

The dataset used in the Stories Clustering project consists of a collection of textual stories. The dataset may vary in size, with a diverse range of topics covered within the stories. Each story is represented as a text document, and the dataset may contain hundreds or thousands of such documents.

The stories in the dataset can be from various sources, including news articles, blog posts, short stories, or user-generated content. The length and complexity of the stories may vary, providing a rich and diverse set of textual data for analysis.

The dataset is preprocessed by removing stopwords, punctuation, and numbers, and by lemmatizing the words to their base form. This preprocessing step helps in reducing noise and improving the accuracy of the topic models.

By analyzing this dataset and clustering the stories based on their topics, we aim to uncover the hidden themes and patterns present in the stories, enabling us to gain valuable insights and facilitate further analysis in various domains, such as content categorization, recommendation systems, or trend analysis.

Overall, the Stories Clustering project combines topic modeling techniques, data preprocessing, and visualization tools to analyze a collection of textual stories and identify the latent topics that exist within the dataset.

Framework

1. **Importing Required Libraries:** The code begins by importing the necessary libraries such as gensim, spacy, matplotlib, sklearn, and keras. These libraries provide various functionalities for text processing, topic modeling, visualization, and machine learning.
2. **Gathering Data:** The code sets the path to the dataset, specifically the Lee corpus and the 20NG dataset. These datasets contain textual stories for analysis. The path is defined using the test_data_dir variable. It also loads the contents of the Lee corpus into the text variable.
3. **Text Preprocessing:** Before proceeding with topic modeling, the code performs text preprocessing. It uses the spacy library to tokenize the text, remove stopwords, punctuation, and numbers, and lemmatize the words. Additionally, it adds custom stopwords to improve the quality of the topic models.
4. **Part-of-Speech (POS) Tagging:** The code demonstrates Part-of-Speech (POS) tagging using the spacy library. It shows how to identify the parts of speech for each word in a given sentence. It also showcases entity recognition, where named entities such as locations or organizations are identified.
5. **Dependency Parsing:** The code showcases dependency parsing using the spacy library. It demonstrates how to extract the grammatical structure and dependencies between words in a sentence. It visualizes the dependency parse tree using the displacy module.
6. **Continuing Cleaning:** After demonstrating POS tagging and dependency parsing, the code continues with the text cleaning process. It removes newline characters and creates a list of lists called texts, where each inner list represents a processed article containing lemmatized words.
7. **Creating Bigrams:** To capture phrases like "New York" as a single entity, the code creates a bigram model using the gensim library. It applies the bigram model to the preprocessed texts, resulting in the detection and joining of relevant bigrams.
8. **Creating Dictionary and Corpus:** The code creates a dictionary and corpus using the preprocessed texts. The dictionary assigns a unique numeric ID to each word, and the corpus represents the texts in a vectorized format where words are replaced with their corresponding IDs and their frequencies.

9. **Topic Modeling - Latent Semantic Indexing (LSI):** The code utilizes the LSI algorithm from the gensim library to perform topic modeling on the corpus. It creates an LSI model with a specified number of topics and displays the top terms associated with each topic.
10. **Topic Modeling - Hierarchical Dirichlet Process (HDP):** The code employs the HDP algorithm from the gensim library to perform topic modeling. HDP automatically determines the number of topics in the corpus and displays the resulting topics.
11. **Topic Modeling - Latent Dirichlet Allocation (LDA):** The code applies the LDA algorithm from the gensim library to perform topic modeling. It creates an LDA model with a specified number of topics and displays the top terms for each topic.
12. **pyLDAvis Visualization:** The code utilizes the pyLDAvis library to visualize the LDA topic model in an interactive manner. It enables the exploration of topics, identification of key terms, and understanding of the topic structure.

By following this detailed outline, one can write the code for the Stories Clustering project. It covers data gathering, text preprocessing, topic modeling using various algorithms, and visualization of the results.

Code Explanation

Importing Required Libraries: The code starts by importing several libraries that will be used throughout the project. These libraries provide powerful tools and functionalities to process and analyze text data. For example, the gensim library is used for topic modeling, spacy is used for natural language processing tasks, matplotlib is used for data visualization, and so on. Think of these libraries as your superhero squad that will assist you in tackling the challenges of text analysis!

Gathering Data: After assembling our superhero squad, it's time to gather the data we'll be working with. In this project, we have two datasets: the Lee corpus and the 20NG dataset. These datasets contain a collection of stories that we will use to discover patterns and clusters. Think of these datasets as treasure chests filled with stories waiting to be explored!

Text Preprocessing: Before we dive into the exciting world of topic modeling, we need to prepare our text data. This step involves cleaning the text, removing unnecessary elements, and transforming it into a suitable format for analysis. The code uses the spacy library to tokenize the text, remove stopwords (common words that don't carry much meaning), punctuation, and numbers. It also lemmatizes the words, which means reducing them to their base or dictionary form. This preprocessing step ensures that we're working with clean and consistent text data.

Part-of-Speech (POS) Tagging: Now that we have our clean text data, let's analyze the parts of speech in the sentences. Imagine each word in a sentence as a character with a specific role to play: nouns, verbs, adjectives, and so on. The code utilizes the spacy library to perform Part-of-Speech (POS) tagging, which assigns a label to each word indicating its grammatical category. This helps us understand the role and function of each word in the sentence.

Dependency Parsing: Let's take our understanding of word relationships to the next level! Dependency parsing helps us uncover the grammatical structure of sentences and identify how words are connected to each other. It's like solving a puzzle where each word has a specific role and depends on other words. The code uses the spacy library to perform dependency parsing, revealing the relationships and dependencies between words. This information can be visualized as a tree-like structure, showcasing the connections between words and their roles in the sentence.

Continuing Cleaning: We've come a long way in understanding the words and their relationships. However, there are still some cleaning tasks left. The code removes newline characters and creates a list of lists called texts, where each inner list represents a processed article containing lemmatized words. This step ensures that our text data is further cleaned and organized for the upcoming analysis.

Creating Bigrams: Sometimes words have more power when they come together! In our case, we want to capture phrases that convey a specific meaning or context. The code creates bigrams, which are pairs of words that frequently occur together, using the gensim library. By identifying and joining these bigrams, we enhance the representation of our text data, allowing us to discover more meaningful topics and patterns.

Creating Dictionary and Corpus: We're almost ready to unleash the power of topic modeling! But first, we need to create a dictionary and corpus. The dictionary assigns a unique numeric ID to each word in our processed text data. The corpus represents the documents in a vectorized format, where words are replaced by their corresponding IDs and their frequencies. These dictionary and corpus structures are essential for training and analyzing our topic models.

Topic Modeling - Latent Semantic Indexing (LSI): The time has come to uncover the hidden topics within our stories! The code employs the Latent Semantic Indexing (LSI) algorithm from the gensim library. LSI analyzes the co-occurrence patterns of words and discovers latent topics that explain the relationships between documents and terms. It creates an LSI model with a specified number of topics and displays the most relevant terms associated with each topic. This allows us to gain insights into the main themes and topics present in our stories.

Topic Modeling - Hierarchical Dirichlet Process (HDP): Now, let's introduce a powerful technique called the Hierarchical Dirichlet Process (HDP) for topic modeling. HDP automatically determines the number of topics present in the corpus, making it a flexible and robust approach. The code applies the HDP algorithm from the gensim library and reveals the resulting topics. With HDP, we let the data guide us in determining the number and content of the topics, providing a data-driven approach to understanding our stories.

Topic Modeling - Latent Dirichlet Allocation (LDA): Our journey through the realm of topic modeling continues with Latent Dirichlet Allocation (LDA). LDA is a popular and

widely used algorithm for discovering topics in text data. The code utilizes the LDA algorithm from the gensim library and creates an LDA model with a specified number of topics. It reveals the top terms associated with each topic, allowing us to grasp the key themes and concepts within our stories.

pyLDAvis Visualization: As we delve deeper into the world of topic modeling, it's important to visualize our results in an interactive and meaningful way. The code employs the pyLDAvis library to create an interactive visualization of the LDA topic model. This visualization allows us to explore the topics, identify the most relevant terms, and understand the overall topic structure of our stories. It's like having a map that guides us through the hidden treasures of knowledge within our text data!

Future Work

So far, we've made significant progress in analyzing and understanding the stories in our dataset. However, there's still more we can do to uncover additional insights and enhance our understanding of the underlying topics and patterns. Here's a step-by-step guide on how to implement the future work:

1. Enhancing Text Preprocessing:

In the current implementation, we performed basic text preprocessing steps like removing stopwords, punctuation, and numbers. To improve the quality of our analysis, we can consider the following enhancements:

- Normalization: Apply techniques like stemming or lemmatization to further normalize the words. This helps in reducing word variations and grouping similar terms together. For example, converting "running," "ran," and "runs" to their base form "run."
- Handling Named Entities: Identify and preserve named entities like person names, locations, or organizations during preprocessing. This can provide valuable insights into specific entities mentioned in the stories.
- Handling Numerical Data: If the dataset contains numerical information, consider how to handle it appropriately. It could involve converting numbers to their textual representation or extracting relevant features from numeric data.

2. Advanced Topic Modeling Techniques:

While LSI, HDP, and LDA are powerful topic modeling algorithms, there are other advanced techniques worth exploring:

- Word Embeddings: Incorporate word embedding models like Word2Vec or GloVe to capture the semantic meaning of words. These models create dense vector representations of words, enabling us to measure word similarities and explore more nuanced topics.
- BERT-Based Models: Utilize transformer-based models like BERT (Bidirectional Encoder Representations from Transformers) to extract contextualized word representations. This allows for a deeper understanding of the context and improves topic modeling accuracy.

- Topic Coherence Evaluation: Measure the coherence and interpretability of the generated topics using metrics like coherence scores. This helps assess the quality of the topics and refine the models accordingly.

3. Optimizing Hyperparameters:

Fine-tuning the hyperparameters of our topic models can significantly impact their performance and the quality of the generated topics. Consider the following steps:

- Grid Search: Perform a grid search to explore different combinations of hyperparameters and evaluate their impact on the topic models' coherence and interpretability. This helps identify the optimal set of hyperparameters.
- Model Evaluation: Assess the performance of topic models using appropriate evaluation metrics. Compare models with varying hyperparameters to identify the best-performing model.

4. Visualizing and Interpreting Results:

Visualization is a powerful tool for understanding and communicating the outcomes of our analysis. Consider the following approaches to visualize and interpret the results:

- Topic Evolution: Analyze how topics evolve over time, if the dataset includes temporal information. Visualize the changes in topic distribution across different time periods to identify trends and patterns.
- Document-Topic Distribution: Visualize the distribution of documents across topics to understand the prevalence and prominence of each topic. Tools like bar plots or stacked area plots can effectively represent this information.
- Interactive Dashboards: Create interactive dashboards using libraries like Plotly or Dash to allow users to explore the topics, drill down into specific documents, and gain a comprehensive understanding of the stories' content.

Step-by-Step Implementation Guide:

Here's a step-by-step guide on how to implement the future work for the Stories Clustering project:

Enhance Text Preprocessing:

- Implement advanced techniques like normalization, named entity recognition, and handling numerical data based on the specific requirements of your dataset.

Explore Advanced Topic Modeling Techniques:

- Incorporate word embeddings or BERT-based models to capture semantic meaning and context.
- Experiment with other topic modeling algorithms like Non-Negative Matrix Factorization (NMF) or Probabilistic Latent Semantic Analysis (pLSA).

Optimize Hyperparameters:

- Perform a grid search to find the best hyperparameter values for your topic models.
- Evaluate models using coherence scores or other relevant metrics.

Visualize and Interpret Results:

- Implement visualizations such as topic evolution over time, document-topic distribution, and interactive dashboards.
- Use libraries like Matplotlib, Plotly, or Seaborn to create visually appealing and informative plots.

By following these steps and continuously iterating on the analysis, you'll be able to uncover deeper insights from your stories and enhance the overall understanding of the corpus.

Concept Explanation

Imagine you have a big bag of stories—some funny, some sad, some mysterious—and you want to organize them into different categories based on their topics. But here's the catch: you don't know the topics in advance, and you don't have time to read each story individually. So how do you go about it? Well, that's where our algorithm comes to the rescue!

The algorithm we used is called "Topic Modeling." It's like having a super-smart robot that reads all the stories, analyzes the words used, and figures out the underlying topics on its own. Isn't that cool?

Now, let's break down how this algorithm works using a simple example. Imagine we have a collection of stories, and we want to discover the main topics hidden within them.

First, we need to prepare our stories for analysis. We remove unnecessary things like stop words (boring words like "the" or "and"), punctuation, and numbers. Then we transform each story into a bag of words, where each word represents a piece of the story's content.

Next, we feed this bag of words to our topic modeling algorithm, which starts working its magic. It examines the patterns in the words and identifies clusters of words that frequently appear together. These clusters represent potential topics hiding within our stories.

To make it more fun, imagine the algorithm as a detective who is trying to solve the mystery of the stories. It looks for clues, like groups of related words, to piece together the topics. Just like Sherlock Holmes deducing the truth from scattered evidence, our algorithm pieces together the topics from the words in the stories.

Once the algorithm has done its job, it presents us with a list of topics and their associated words. It's like opening a treasure chest and finding different gems representing the topics in our stories. For example, we might discover topics like "adventures and travel" or "love and relationships."

But wait, there's more! We can even visualize these topics using fancy graphs and charts. It's like creating a beautiful map of our story universe, where each topic is a distinct land with its own unique characteristics.

So, by using topic modeling, we can unravel the hidden themes in our stories without manually reading each one. It's like having a smart assistant who understands the stories' essence and organizes them into meaningful clusters.

Now, armed with this knowledge, you can embark on your own storytelling adventure and unlock the secrets hidden within the words. Happy topic modeling, my friend!

Exercise Questions

1. Question: What is the purpose of preprocessing the text data in the code provided?

Answer: The purpose of preprocessing the text data is to clean and prepare it for topic modeling. The preprocessing steps involve removing stopwords (common words with little meaning), punctuation, numbers, and newline characters. Additionally, the text is lemmatized (reduced to its base form) to ensure consistent representation. These preprocessing steps help to improve the accuracy and effectiveness of the topic modeling algorithm by removing noise and irrelevant information from the text data.

2. Question: What is the significance of using bigrams in the code provided?

Answer: Bigrams are two consecutive words that often appear together. By detecting and joining bigrams, we can capture important phrases or combinations of words that carry specific meanings. In the code, a Phrases model is created, which detects and joins bigrams in the preprocessed text data. This helps to enhance the representation of the text data by preserving meaningful word combinations. For example, "New York" as a bigram captures the entity "New York" as a single term rather than treating "New" and "York" separately.

3. Question: Explain the purpose of topic modeling using algorithms like LSI, HDP, and LDA.

Answer: The purpose of topic modeling is to automatically discover underlying topics or themes within a collection of documents. Algorithms like Latent Semantic Indexing (LSI), Hierarchical Dirichlet Process (HDP), and Latent Dirichlet Allocation (LDA) are commonly used for topic modeling.

LSI: LSI decomposes the matrix of words in the document collection to identify key topics. It represents documents and topics as vectors in a lower-dimensional semantic space, allowing for efficient similarity comparisons and topic extraction.

HDP: HDP is an unsupervised topic model that determines the number of topics present in the document collection automatically. It infers the topic proportions and word distributions for each document without requiring prior knowledge of the number of topics.

LDA: LDA is a popular topic modeling algorithm that assumes documents are mixtures of topics, and topics are distributions over words. It estimates the topic proportions for each document and the word distributions for each topic, providing a meaningful representation of the document collection.

These algorithms help uncover hidden patterns and structures within the text data, enabling us to gain insights into the main themes or topics discussed in the documents.

4. Question: What is the role of pyLDAvis in the project code?

Answer: pyLDAvis is a Python library used for interactive topic model visualization. In the project code, pyLDAvis is utilized to visualize the LDA topic model. It extracts information from the fitted LDA model, such as topic-word distributions and document-topic proportions, and presents it in an interactive web-based visualization. This visualization allows users to explore and interpret the topics discovered by the model, making it easier to understand and analyze the content of the document collection.

5. Question: How can topic modeling be useful in real-world applications?

Answer: Topic modeling has various real-world applications across different domains:

- Content categorization: Topic modeling can automatically categorize large volumes of text data into meaningful topics, enabling efficient content organization and retrieval.
- Recommender systems: By understanding the topics discussed in documents or user preferences, topic modeling can enhance recommender systems by suggesting relevant content or products.
- Market research: Topic modeling can help analyze customer feedback, reviews, and social media data to gain insights into customer opinions, preferences, and emerging trends.
- News and media analysis: Topic modeling can assist in analyzing news articles, blog posts, and social media discussions to identify popular topics, track public sentiment, or summarize news events.
- Medical research: Topic modeling can be applied to analyze medical literature and identify key research themes or trends, aiding in literature review and knowledge discovery.