LDA

王铭泽 ZY2203115

wmz20000729@buaa.edu.cn

1 Introduction

The Latent Dirichlet Allocation (LDA) model has widespread applications in fields such as text mining, information retrieval, and recommender systems. In this report, 200 paragraphs (each containing more than 500 words) were uniformly sampled from the corpus provided in the link. The label for each paragraph corresponds to the novel it belongs to. We used the LDA model to build a topic representation of the text, representing each paragraph with a topic distribution for classification. The classification results were then evaluated and analyzed.

2 Methods

The Latent Dirichlet Allocation (LDA) is a topic model that can be used for text modeling and analysis. Topic models are unsupervised learning algorithms aimed at learning the topic structure from text data, thereby identifying the hidden semantic structures and topic information within the text.

The basic idea of the LDA model is to assume that each document is composed of a mixture of several topics, and each topic is a mixture of several words. Specifically, for a set of documents, the LDA model considers the documents to be composed of several topics, with each topic represented by a fixed word distribution, i.e., the words in each topic appear with certain probabilities. The goal of the LDA model is to learn the topic distribution and word distribution in the document collection, which can then be used for topic inference and text classification of new text.

The process of using the LDA model for text modeling is as follows:

Data preprocessing: Preprocess the text data, including removing stopwords, performing stemming, and removing low-frequency words.

Building the bag-of-words model: Convert the text into a vector representation, which is usually done by creating a vocabulary of words in the text and representing each text as a vector where each dimension corresponds to the frequency of a word.

Training the LDA model: Use the constructed bag-of-words model to train the LDA model, learning the topic distribution and word distribution in each topic. When training the LDA model, parameters such as the number of topics and hyperparameters need to be specified.

Inferring topic distribution for text: For new text, the trained LDA model can infer the topic distribution, i.e., the probability distribution of each topic in the text.

Topic classification: Based on the topic distribution of the text, the text can be classified into topics, assigning it to the most probable topic category.

3 Experiment

In the specific experimental part, we uniformly extracted 200 paragraphs (each containing more than 500 words) from the given corpus, with each paragraph's label corresponding to the novel it belongs to. We used the LDA model for text modeling and represented each paragraph as a topic distribution for classification.

During the data preprocessing stage, we were required to uniformly extract 200 paragraphs (each containing more than 500 words) with labels that correspond to the novel they belong to. Analyzing the given corpus revealed that it contained a total of 16 articles. We extracted 13 paragraphs from each article, resulting in 208 paragraphs. After splitting words for each article, we divided the total number of words in an article by 13, giving us 13 intervals. We selected paragraphs from each interval by extracting the first 500 words.

Python's LDA topic model can perform a variety of operations, such as outputting the top-N highest frequency words in each dataset; outputting the weights and topics corresponding to each word in the articles; outputting the distribution probability of articles and topics (each row in the text represents an article with probabilities indicating the likelihood of the article belonging to a specific topic); and outputting the distribution probability of characteristic words and topics, forming a K*M matrix where K is the number of categories set and M is the total number of words in all articles.

During the model training stage, based on the LDA principle, we initialized by randomly assigning an initial Topic value to each word in every article. We then counted the total number of words in each article, each article's word frequency, each topic's total word count, and each topic's word frequency. We calculated the probability of each topic being selected and then proceeded to train the model through iterations.

In the model testing stage, we still selected paragraphs from the corresponding 16 novels as the testing set. In the training set, we chose the 0-500th words among the 208 paragraphs. Therefore, in the testing set, we selected the 501st-1000th words to form the test paragraphs, ultimately generating the articles to be classified. We preprocessed the 15 articles to be classified after reading them and randomly assigned an initial Topic to each word in the 15 articles, similarly counting each article's total word count and word frequency, recording the results. However, at this stage, we no longer needed to count each topic's total word count and word frequency, as they had already been determined during model training and were treated as known quantities during testing.

Next, we needed to determine which novel each article to be classified originated from based on the probability results. We used the Euclidean distance method to compare the probability vectors of each Topic between the articles to be classified and the known novels, identifying the closest novel as the source of the articles.

4 Conclusion

In this experiment, we used the LDA model to classify the topics of Jin Yong's novels. The experimental results show that, with an appropriate number of topics, the LDA model can effectively classify novel paragraphs. This assignment has deepened my understanding of the applications of natural language processing. However, there are still some shortcomings in this task. In data analysis, it's typical to evaluate the quality of an algorithm using measures such as accuracy, recall, or F-score. Researchers also continuously optimize models or replace them with better algorithms.