The Second Report of Deep Learning for Natural Language Processing

Yue Zhao
zy2303607@buaa.edu.cn

Abstract

This report presents a comprehensive study on topic modeling using Latent Dirichlet Allocation (LDA) and its application in document classification. We explore the effectiveness of Euclidean Distance and Support Vector Machine (SVM) classifiers based on the topic distributions obtained from LDA. The study involves preprocessing Chinese corpus, implementing LDA, and evaluating classification performance through K-Fold cross-validation. The results highlight the superior performance of SVM over Euclidean Distance in terms of classification accuracy across various parameter settings.

Introduction

Topic Model: In a document about feline animals, different sections might discuss cats, tigers, and cheetahs, with each section featuring a higher frequency of associated words related to that topics—for instance, "fish" and "mouse" for cats, "king of the forest" for tigers, and "spots" for cheetahs. This indicates that a document can contain multiple topics, each with a different proportion of associated words. Topic models use mathematical frameworks to capture these characteristics by analyzing the frequency of words within documents to determine the present topics and their respective proportions. Essentially, topic models focus on learning two distributions: the document-topic distribution and the topic-word distribution. One of the common topic models is LDA.

LDA Model: Latent Dirichlet Allocation (LDA) is a generative model for document topics that encompasses a three-tier structure of words, topics, and documents, also known as a three-layer Bayesian probability model. The LDA model posits that each document represents a probability distribution of various topics, and each topic, in turn, represents a probability distribution of various words. Therefore, every word in a document is generated through a process of "selecting a topic with a certain probability and choosing a word from this topic with a certain probability," allowing each document to be viewed as a frequency vector of words. This transforms textual information into numerical data that is more amenable to modeling. The document-to-topic distribution follows a multinomial distribution, denoted as θ; similarly, the topic-to-word distribution also follows a multinomial distribution, denoted as φ, as shown in **Figure 1**.

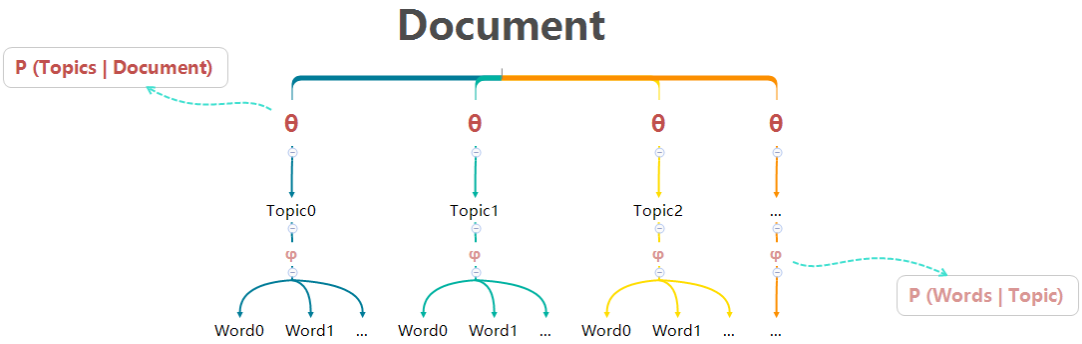

Figure 1: The flowchart of LDA Model Generating Words

As an unsupervised machine learning technique, LDA is utilized to uncover latent thematic structures within large sets of documents or corpora. However, LDA's method of generating documents does not account for the sequential order of words, which simplifies the complexity of the problem but also presents opportunities for model enhancement.

After obtaining the probability distribution of different documents under different topics using the LDA model, it can be used as a feature for document classification. Two of the commonly used classifiers are Euclidean distance and SVM.

Euclidean Distance: The Euclidean distance classification method is a distance-based classification technique commonly employed in various machine learning and pattern recognition tasks. The fundamental concept of this method involves determining the similarity between data points by calculating the Euclidean distances between them. Its main advantage lies in its simplicity and ease of implementation. However, it tends to perform poorly with high-dimensional data and is sensitive to noise and outliers.

SVM: Support Vector Machine (SVM) is a type of generalized linear classifier that performs binary classification on data in a supervised learning manner. The decision boundary of SVM is the maximum-margin hyperplane derived from the training samples. SVM employs the hinge loss function to calculate empirical risk and incorporates regularization terms in the solving system to optimize structural risk, making it a classifier characterized by sparsity and robustness. Additionally, SVM can perform non-linear classification through kernel methods, making it one of the commonly used kernel learning techniques.

Methodology

M1: LDA Model

The overall process of LDA is as follows:

- Define a set of documents D and a set of topics T.
- Each document d in D is considered as a sequence of words $\langle w_1, w_2, ..., w_n \rangle$, where w_i represents the i-th word and d contains n words. (In LDA, this is referred to as a "word bag"; the actual positions of the words do not affect the algorithm.)

- All distinct words involved in D constitute a large set called $VOCABULARY$. LDA takes the document set D as input and aims to train two resulting vectors, assuming there are k topics and the $VOCABULARY$ contains m words.
- For each document d in D, the probability θ_d corresponding to different topics is $\langle p_{t_1}, p_{t_2}, ..., p_{t_k} \rangle$ where p_{t_i} denotes the probability of document d corresponding to the i-th topic in T. The method of calculation is intuitive as **Formula 1**:

$$p_{t_i} = \frac{n_{t_i}}{n} \tag{1}$$

, where n_{t_i} represents the number of words in d that correspond to the i-th topic, and n is the total number of words in d.

- For each topic t in T, the probability φ_t of generating different words is $\langle p_{w_1}, p_{w_2}, ..., p_{w_m} \rangle$, where p_{w_i} represents the probability of t generating the i-th word in $VOCABULARY$. The method of calculation is similarly straightforward as **Formula 2**:

$$p_{w_i} = \frac{N_{w_i}}{N} \tag{2}$$

, where N_{w_i} represents the count of the i-th word in $VOCABULARY$ corresponding to topic t, and N is the total count of all words corresponding to topic t.

- The core formula of LDA is **Formula 3**:

$$p(w|d) = p(w|t) \times p(t|d) \tag{3}$$

Here, $p(t|d)$ is calculated using θ_d and $p(w|t)$ using φ_t.

Intuitively, the **Formula 3** uses the topic as an intermediary layer, providing the probability of a word w appearing in document d based on the current θ_d and φ_t. In practice, using the current θ_d and φ_t, we can calculate the probability $p(w|d)$ for a word in a document corresponding to any topic, and then use these results to update the topic corresponding to the word. If this update changes the topic associated with the word, it will in turn affect θ_d and φ_t.

M2: Classifier

Based on the probability outcomes derived from the Latent Dirichlet Allocation (LDA) model, each document to be classified is distinguished by identifying its source novel.

If employing the **Euclidean distance** method, it is necessary to compare the document under classification (i.e., the test set document) with known documents in the training set. When the distance between their respective topic probability vectors is the smallest, it is assumed that their labels are consistent, indicating that the current document under classification originates from the novel corresponding to the compared document in the training set.

For two points $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$ in an n-dimensional space, the Euclidean distance between them is calculated using **Formula 4**:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2} \quad (4)$$

If employing the **Support Vector Machine (SVM)** method, its fundamental concept is to construct a hyperplane that maximizes the margin between different classes of data, where the margin is defined as the distance from the nearest data points (known as support vectors) to the hyperplane. In practice, many datasets are not linearly separable. SVM addresses this by employing kernel functions to map the original data into a higher-dimensional space where the data can be linearly separated. Commonly used kernel functions include linear, polynomial, and Radial Basis Function (RBF) kernels. The problem of finding the maximum margin is transformed into an optimization problem, typically solved by addressing a convex quadratic programming problem.

Experimental Studies

E1: Preprocessing of Chinese Corpus

The preprocessing of Chinese corpus primarily involves four steps: data loading, exclusion, tokenization, and sampling.

Initially, the raw Chinese corpus is loaded into the program. Subsequent steps involve removing meaningless characters, including stopwords, punctuation, letters, and advertisements etc. The data set at this stage is then segmented into tokens, which can be characters or words, resulting in a tokenized data set used for paragraph sampling.

According to the requirements, 1000 paragraphs are to be uniformly extracted, with each paragraph containing K tokens. Analysis of the given corpus, which includes 16 novels, indicates that it is necessary to extract approximately 63 paragraphs from each novel, totaling 1008 paragraphs. After tokenizing each novel by characters or words, the total number of tokens in a novel is divided by 63, establishing 63 intervals. The paragraphs selected for extraction correspond to the first K tokens within each interval.

E2: Results of LDA & Classifier

Based on the preprocessed dataset, ten different training and testing sets are derived using K-Fold cross-validation to perform ten rounds of cross-validation. Each round of cross-validation involves initial iterative training and testing using the Latent Dirichlet Allocation (LDA) model,

followed by classification using Euclidean distance and Support Vector Machine (SVM) based on the probability outputs from LDA, as shown in **Figure 2**. The method of controlled variable is used to adjust various parameters to study their impact on the final classification outcomes.

```
[[ECHO_CROSS]]:  9
========= MODEL TRAINING =========
Final Output of Train:
Doc_pronew:  [[0.00433478 0.01267089 0.01533845 ... 0.00966989 0.00633545 0.00366789]
 [0.00433478 0.01100367 0.01367122 ... 0.00933645 0.00666889 0.00333444]
 [0.006002   0.00833611 0.01067022 ... 0.00966989 0.00700233 0.00233411]
 ...
 [0.00866956 0.015005   0.01133711 ... 0.012004   0.01000333 0.01600534]
 [0.009003   0.01467156 0.01033678 ... 0.01267089 0.01000333 0.01567189]
 [0.00866956 0.01467156 0.01100367 ... 0.01400467 0.00833611 0.01533845]]
Number of loop_train:  6
========= Model training completed!  =========
========= MODEL TESTING =========
Final Output of Test:
Doc_pronew:  [[0.00433478 0.01267089 0.01533845 ... 0.00966989 0.00633545 0.00366789]
 [0.00433478 0.01100367 0.01367122 ... 0.00933645 0.00666889 0.00333444]
 [0.006002   0.00833611 0.01067022 ... 0.00966989 0.00700233 0.00233411]
 ...
 [0.00866956 0.015005   0.01133711 ... 0.012004   0.01000333 0.01600534]
 [0.009003   0.01467156 0.01033678 ... 0.01267089 0.01000333 0.01567189]
 [0.00866956 0.01467156 0.01100367 ... 0.01400467 0.00833611 0.01533845]]
Doc_pronew_test:  [[0.01100367 0.01633878 0.01133711 ... 0.01400467 0.00800267 0.01433811]
 [0.01300433 0.01667222 0.01033678 ... 0.01333778 0.00700233 0.01367122]
 [0.01533845 0.01733911 0.01000333 ... 0.01400467 0.00566856 0.01300433]
 ...
 [0.10836946 0.00933645 0.00700233 ... 0.00866956 0.02834278 0.03201067]
 [0.12137379 0.01067022 0.00933645 ... 0.00833611 0.03867956 0.01000333]
 [0.13304435 0.00333444 0.00733578 ... 0.01100367 0.04301434 0.01333778]]
Number of loop_test:  4
========= Model testing completed!  =========
```

Figure 2: Example of the final iteration results of LDA model training and testing in a certain cross validation round

For analysis based on **words as units (chr=0)**, keep the **Tokens = 500,** while **Topics = 20, 50, 80, 110, and 140,** with results as shown in **Figure 3-7**:

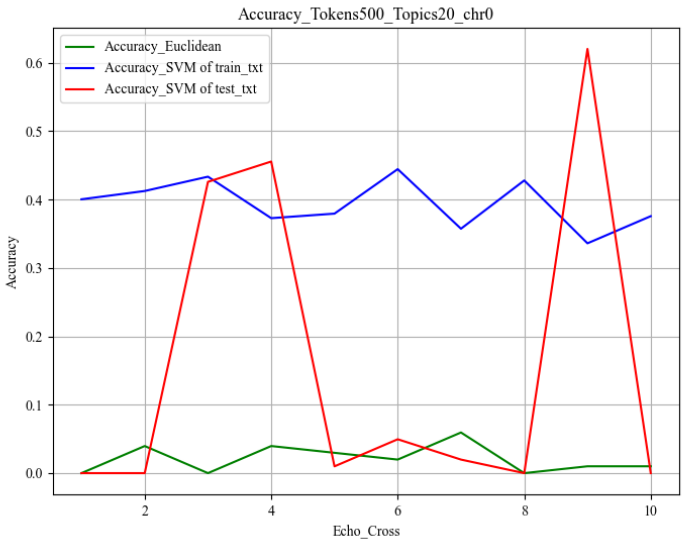

Figure 3: The accuracy of different cross validation echos (Tokens = 500, **Topics = 20**, chr=0)

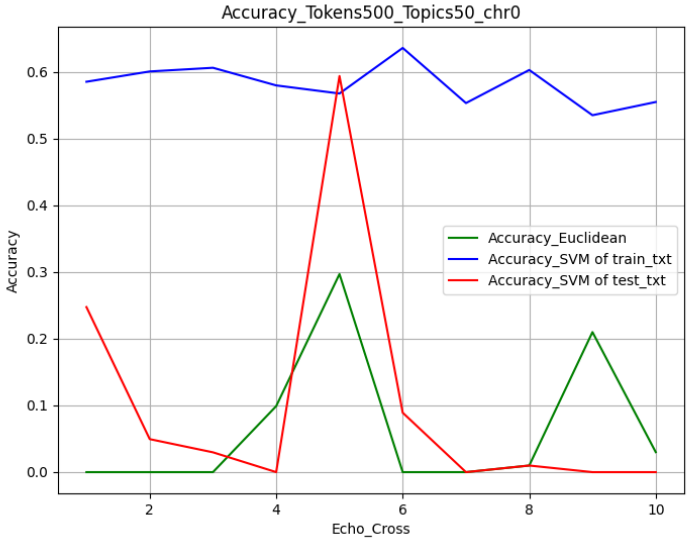

Figure 4: The accuracy of different cross validation echos (Tokens = 500, **Topics = 50**, chr=0)

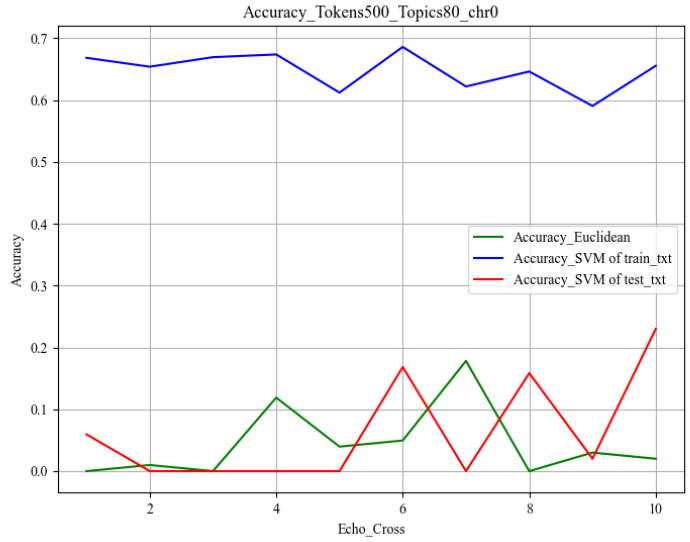

Figure 5: The accuracy of different cross validation echos (Tokens = 500, **Topics = 80**, chr=0)

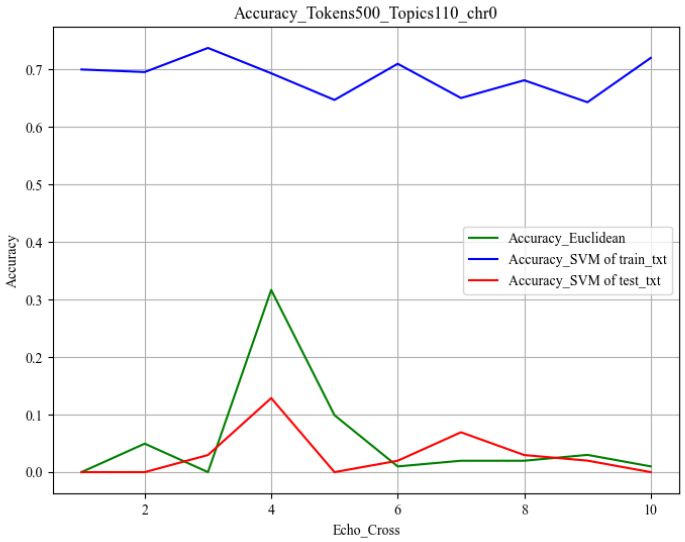

Figure 6: The accuracy of different cross validation echos (Tokens = 500, **Topics = 110**, chr=0)

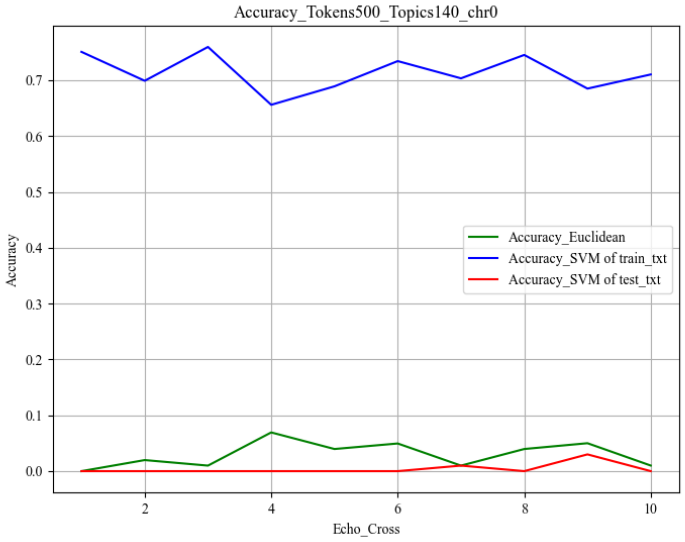

Figure 7: The accuracy of different cross validation echos (Tokens = 500, **Topics = 140**, chr=0)

As can be observed from **Figure 3 - 7**, as the Topics increases, there is a general upward trend in the accuracy of the SVM classification results (represented by the **blue line**). In contrast, the classification accuracy using Euclidean distance (represented by the green line) remains significantly lower than that achieved by SVM.

For analysis based on **words as units**, keep the **Topics = 80,** while **Tokens = 20, 100, 500, 1000, and 3000,** with results as shown in **Figure 8-12**:

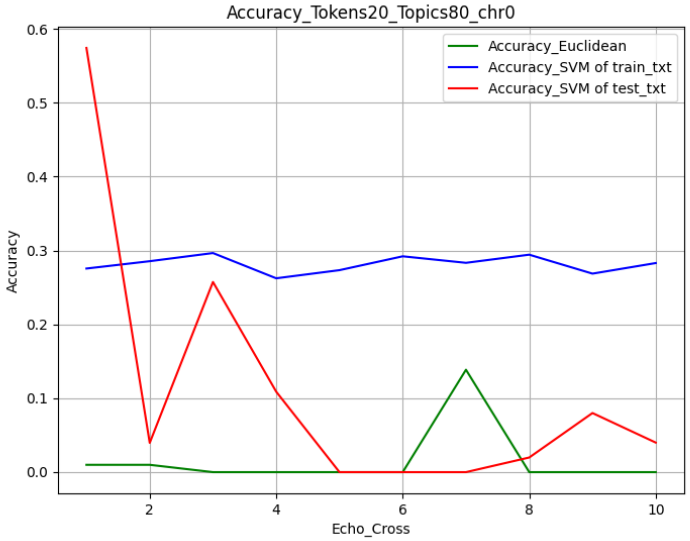

Figure 8: The accuracy of different cross validation echos (**Tokens = 20**, Topics = 80, chr=0)

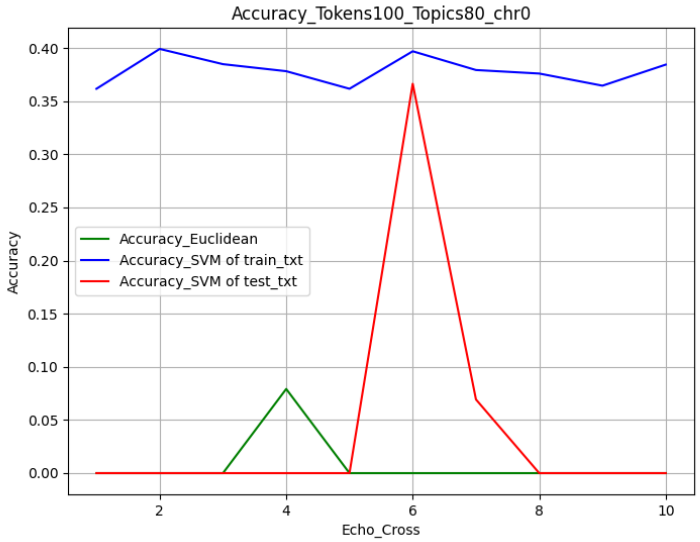

Figure 9: The accuracy of different cross validation echos (**Tokens = 100**, Topics = 80, chr=0)

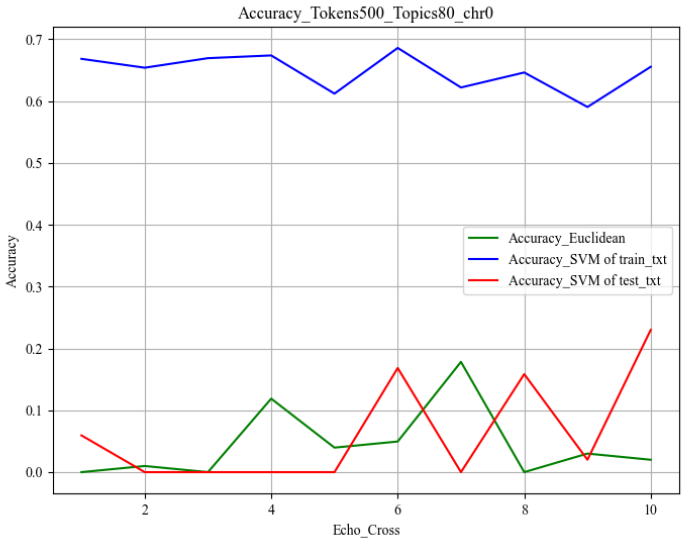

Figure 10: The accuracy of different cross validation echos (**Tokens = 500**, Topics = 80, chr=0)

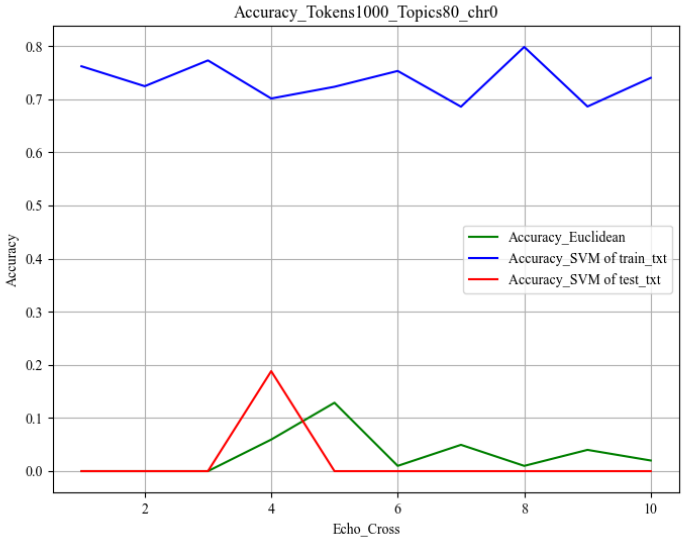

Figure 11: The accuracy of different cross validation echos (**Tokens = 1000**, Topics = 80, chr=0)

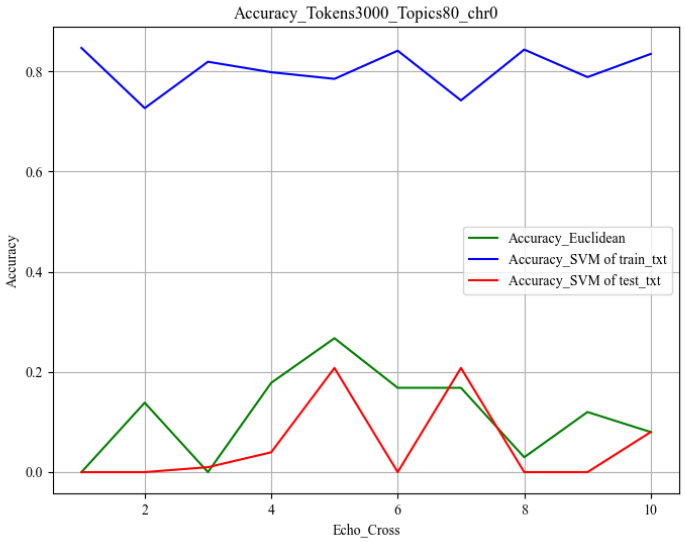

Figure 12: The accuracy of different cross validation echos
(**Tokens = 3000**, Topics = 80, chr=0)

As can be observed from **Figure 8 - 12**, as the Tokens increases, there is a general upward trend in the accuracy of the SVM classification results (represented by the **blue line**). In contrast, the classification accuracy using Euclidean distance (represented by the green line) remains significantly lower than that achieved by SVM.

Maintaining the **Tokens = 500** and **Topics = 80**, but switching to **characters as units,** the results are as shown in **Figure 13**:

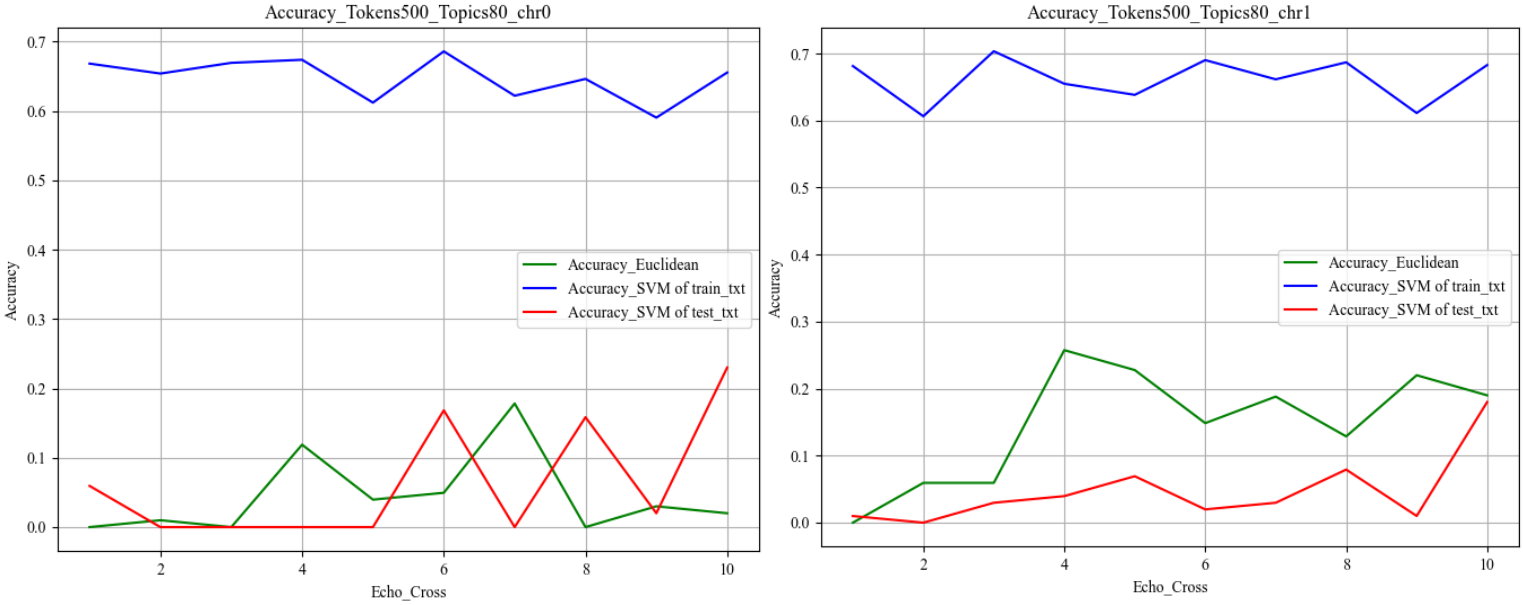

Figure 13: The accuracy of different cross validation echos
(**left:** Tokens = 500, Topics = 80, **chr=0**; **right:** Tokens = 500, Topics = 80, **chr=1**)

As can be observed from **Figure 13,** when the basic unit of tokenization is changed, the classification accuracy of SVM (indicated by the **blue line**) shows no significant variation, while

the classification accuracy using Euclidean distance (indicated by the **green line**) experiences a slight increase. However, overall, under the current parameters, switching to characters as the basic unit for tokenization has a minimal impact on the final classification results.

Conclusions

In conclusion, the use of LDA for topic modeling combined with SVM for classification significantly outperforms the Euclidean Distance classifier. The experimental results demonstrate that increasing the number of topics and tokens generally enhances the classification accuracy. Additionally, the choice of tokenization unit (words vs. characters) has a minimal impact on SVM performance but slightly affects the Euclidean Distance classifier. This study underscores the robustness and efficiency of SVM in handling topic-based document classification tasks.

References

[1] https://blog.csdn.net/xiaohutong1991/article/details/107924703