Report of Deep Learning for Natural Langauge Processing

Shanghua Li
zy2403112@buaa.edu.com

Abstract

This report explores the effectiveness of using Latent Dirichlet Allocation (LDA) topic modeling for text classification on Jin Yong's novels. The experiment uses paragraphs of different lengths (K=20, 100, 500, 1000, 3000) as input data and applies different numbers of topics (T=5, 10, 20, 50) for topic modeling. Each paragraph is represented as a topic distribution and classified using a Naïve Bayes classifier, with 10-fold cross-validation used to evaluate classification performance. This report primarily analyzes: (1) whether the number of topics affects classification performance; (2) the impact of using "words" versus "characters" as basic units on classification results; (3) how short and long paragraphs (different K values) influence topic modeling performance.

Introduction

Text classification is an essential task in Natural Language Processing (NLP), and topic modeling (e.g., LDA) can effectively represent document themes for classification purposes. This study applies LDA-based text representation and classification to paragraphs from Jin Yong's novels.

Key research questions:

Does changing the number of topics (T) affect classification performance?

Does using "words" or "characters" as the basic unit influence classification accuracy?

How does paragraph length (K=20 vs. K=3000) impact the performance of the LDA model?

Methodology

Data Processing

Data Source: Jin Yong's novels

Preprocessing Steps:

Load the text and use jieba for Chinese word segmentation (if USE_WORDS=True); otherwise, analyze at the character level

Split the text into paragraphs of different lengths (K values), each labeled with its corresponding novel

Topic Modeling

LDA (Latent Dirichlet Allocation) is applied to extract topics

Text is converted into a Bag-of-Words model using CountVectorizer before LDA training

Experiments are conducted with different topic numbers T = [5, 10, 20, 50]

Classification Task

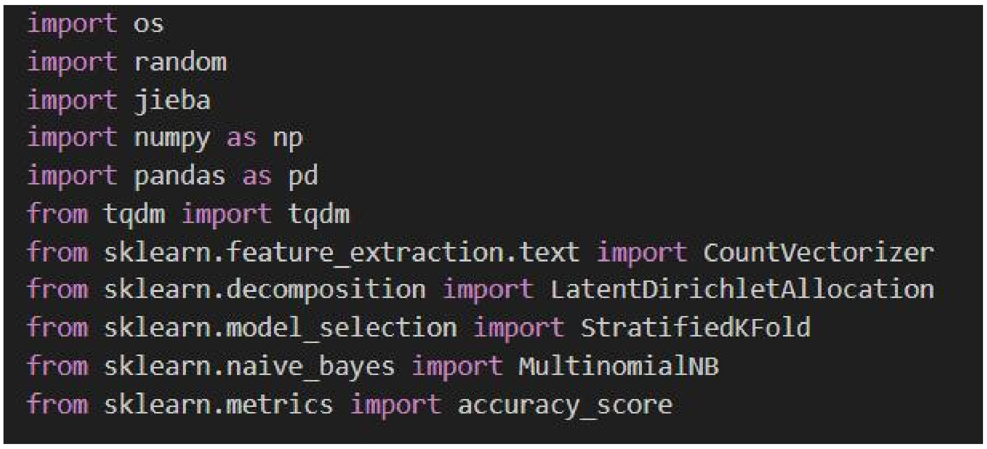
A Naïve Bayes (MultinomialNB) classifier is used

10-fold cross-validation is applied (900 samples for training, 100 for testing).

Experimental Studies

This Python code implements LDA topic modeling combined with a Naïve Bayes classifier for Jin Yong novel classification. The workflow is as follows:

Importing Required Libraries



```
import os
import random
import jieba
import numpy as np
import pandas as pd
from tqdm import tqdm
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.model_selection import StratifiedKFold
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
```

Jieba : Tokenizing Chinese text during preprocessing.

Numpy : Processing numerical features or matrix operations (e.g., TF-IDF matrices).

pandas(abbreviated as `pd`) : Loading, cleaning, and structuring text data (e.g., from CSV files).

Tqdm : Displaying progress during data preprocessing or model training.

sklearn.feature_extraction.text.CountVectorizer: Feature extraction for text classification tasks.

sklearn.decomposition.LatentDirichletAllocation(LDA):Topic discovery in news classification or sentiment analysis.

sklearn.model_selection.StratifiedKFold: Evaluating model performance on small or imbalanced datasets.

sklearn.naive_bayes.MultinomialNB : Spam email classification or sentiment analysis.

sklearn.metrics.accuracy_score : Evaluating model performance on a test set.

Overview of the Code

Objective: Extract text features using LDA topic modeling and classify them using a Naïve Bayes classifier.

Data: The complete works of Jin Yong, where each novel is a separate file stored in the `FOLDER_PATH` directory.

Methods:

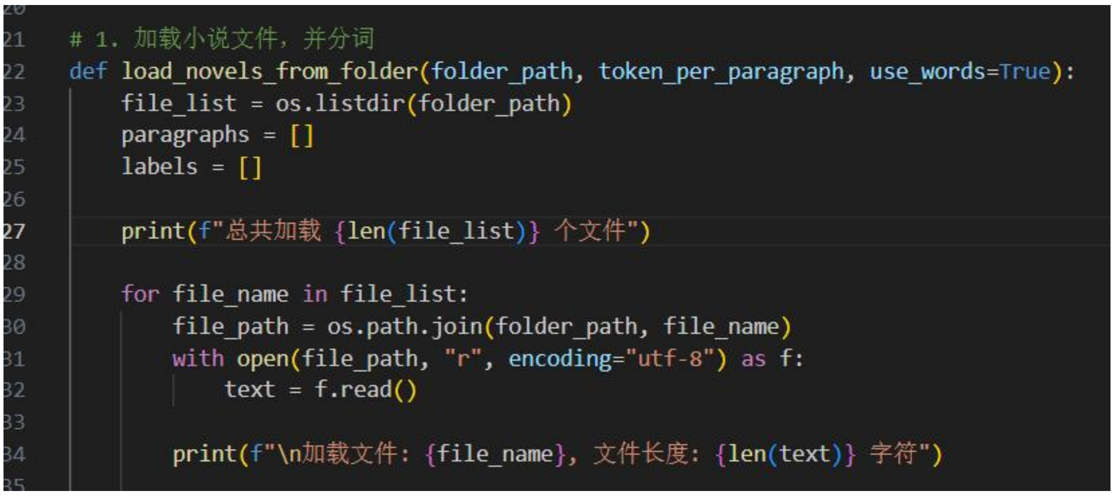
Preprocessing: Load text data and split it into fixed-length segments (using characters or words).

Feature Extraction: Use CountVectorizer to extract word frequency features and apply LatentDirichletAllocation (LDA) to reduce dimensions.

Classification: Use Naïve Bayes for text classification and evaluate model performance using 10-fold cross-validation.

Detailed Code Explanation

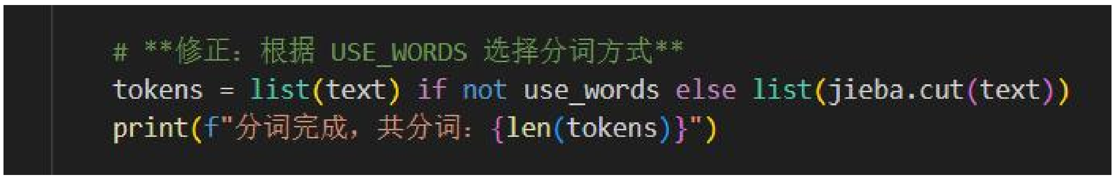
Loading Novel Files and Tokenization



```python
# 1. 加载小说文件，并分词
def load_novels_from_folder(folder_path, token_per_paragraph, use_words=True):
    file_list = os.listdir(folder_path)
    paragraphs = []
    labels = []

    print(f"总共加载 {len(file_list)} 个文件")

    for file_name in file_list:
        file_path = os.path.join(folder_path, file_name)
        with open(file_path, "r", encoding="utf-8") as f:
            text = f.read()

        print(f"\n加载文件: {file_name}, 文件长度: {len(text)} 字符")
```

This function reads Jin Yong's novels from folder_path, performs tokenization, and splits the text into paragraphs of fixed length.

Choice of Tokenization Method

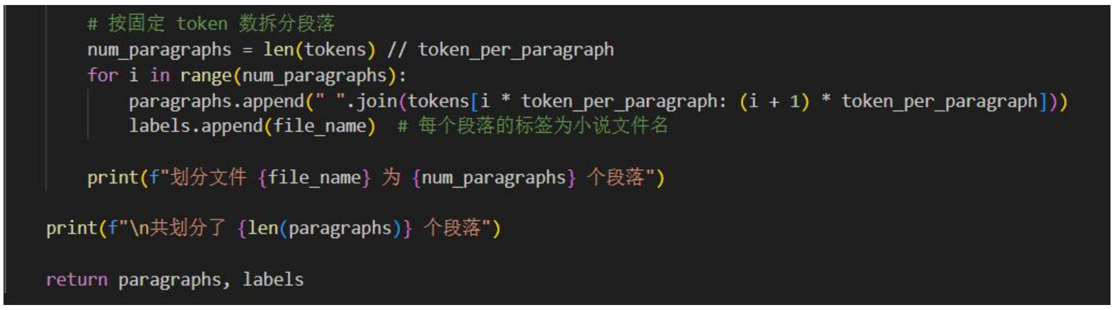


```python
        # **修正: 根据 USE_WORDS 选择分词方式**
        tokens = list(text) if not use_words else list(jieba.cut(text))
        print(f"分词完成，共分词: {len(tokens)}")
```

If USE_WORDS=False, the text is split character by character.

If USE_WORDS=True, the text is tokenized using jieba.cut.

Splitting the Text into Fixed-Length Paragraphs

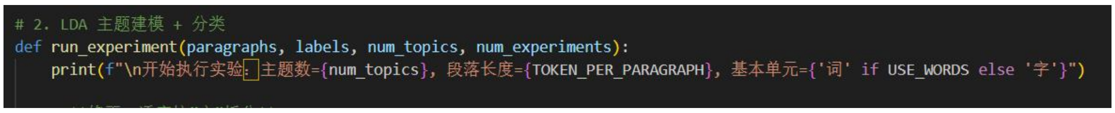


```python
        # 按固定 token 数拆分段落
        num_paragraphs = len(tokens) // token_per_paragraph
        for i in range(num_paragraphs):
            paragraphs.append(" ".join(tokens[i * token_per_paragraph: (i + 1) * token_per_paragraph]))
            labels.append(file_name)  # 每个段落的标签为小说文件名

        print(f"划分文件 {file_name} 为 {num_paragraphs} 个段落")

    print(f"\n共划分了 {len(paragraphs)} 个段落")

    return paragraphs, labels
```

Each paragraph contains token_per_paragraph tokens.

Each paragraph's label is the corresponding novel's filename (used for classification).

LDA Topic Modeling + Naïve Bayes Classification

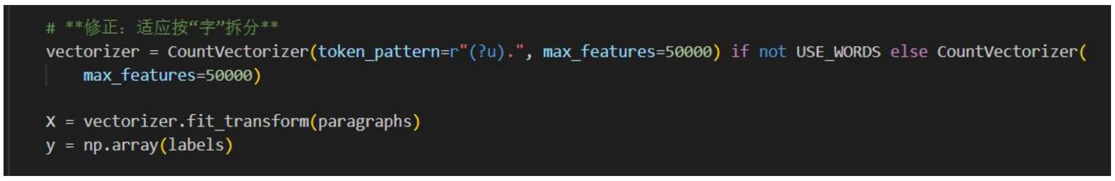


```python
# 2. LDA 主题建模 + 分类
def run_experiment(paragraphs, labels, num_topics, num_experiments):
    print(f"\n开始执行实验：主题数={num_topics}, 段落长度={TOKEN_PER_PARAGRAPH}, 基本单元={'词' if USE_WORDS else '字'}")
```

This function extracts features using CountVectorizer, reduces dimensionality using LatentDirichletAllocation (LDA), and classifies the text using a Naïve Bayes classifier.

It uses 10-fold cross-validation to evaluate classification accuracy.

Feature Extraction (CountVectorizer)

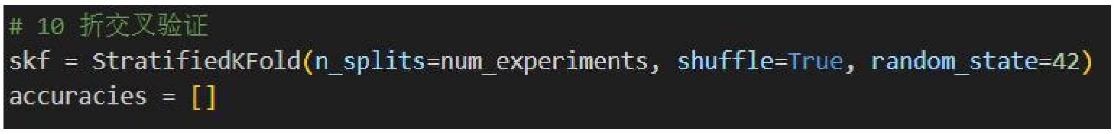


```python
    # **修正：适应按“字”拆分**
    vectorizer = CountVectorizer(token_pattern=r"(?u).", max_features=50000) if not USE_WORDS else CountVectorizer(
        max_features=50000)

    X = vectorizer.fit_transform(paragraphs)
    y = np.array(labels)
```

token_pattern=r"(?u)." ensures character-level tokenization when USE_WORDS=False.

max_features=50000 limits the vocabulary size.

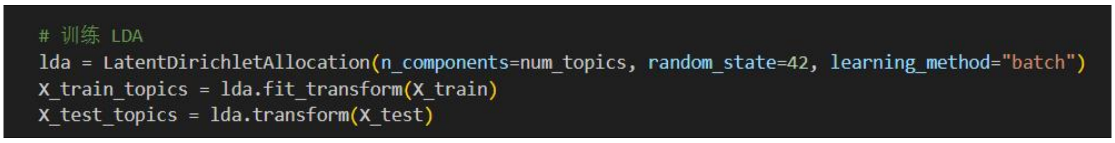
10-Fold Cross-Validation



```python
# 10 折交叉验证
skf = StratifiedKFold(n_splits=num_experiments, shuffle=True, random_state=42)
accuracies = []
```

StratifiedKFold ensures balanced class distribution across folds.

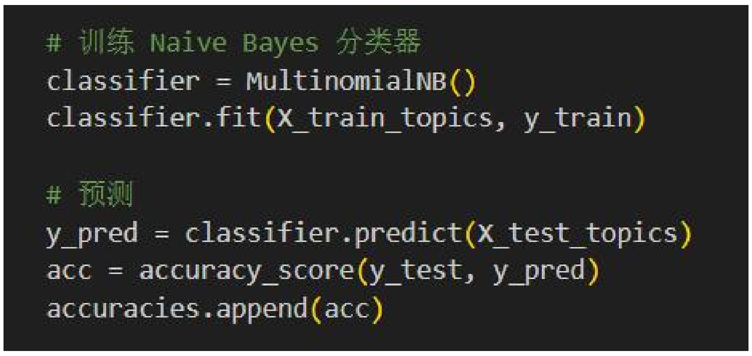
LDA Topic Modeling



```python
# 训练 LDA
lda = LatentDirichletAllocation(n_components=num_topics, random_state=42, learning_method="batch")
X_train_topics = lda.fit_transform(X_train)
X_test_topics = lda.transform(X_test)
```

LDA reduces the high-dimensional word frequency matrix to a num_topics-dimensional topic representation.

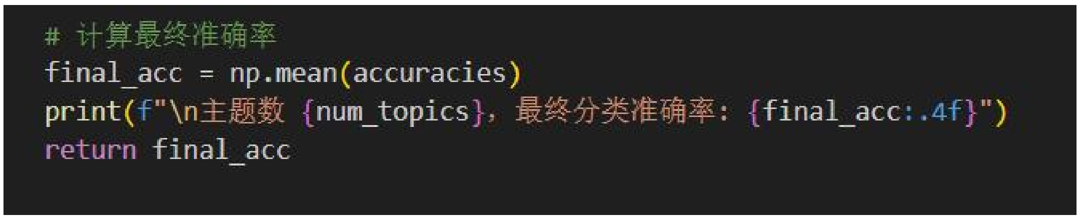
Naïve Bayes Classification



```python
# 训练 Naive Bayes 分类器
classifier = MultinomialNB()
classifier.fit(X_train_topics, y_train)

# 预测
y_pred = classifier.predict(X_test_topics)
acc = accuracy_score(y_test, y_pred)
accuracies.append(acc)
```

A Multinomial Naïve Bayes classifier is trained and tested.

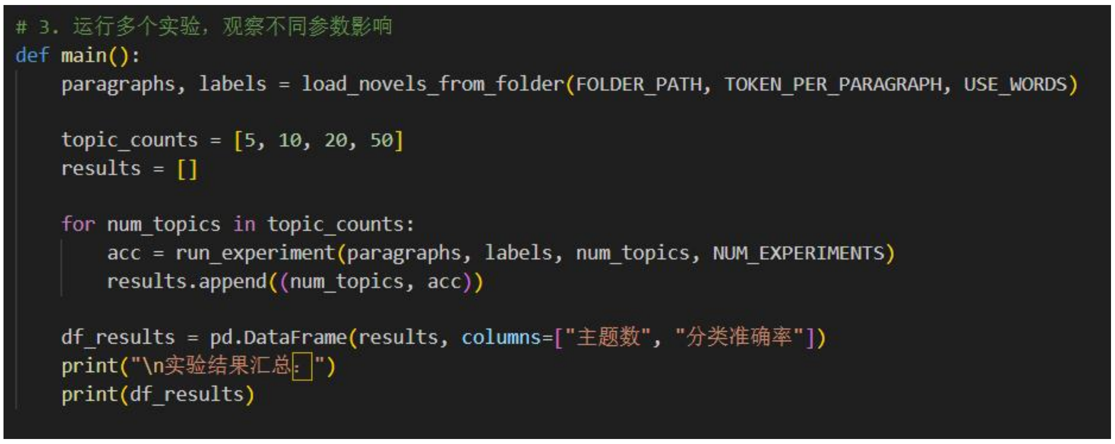
Final Accuracy Calculation



```python
# 计算最终准确率
final_acc = np.mean(accuracies)
print(f"\n主题数 {num_topics}, 最终分类准确率: {final_acc:.4f}")
return final_acc
```

The average accuracy across 10 folds is computed.

Running Multiple Experiments with Different Topic Counts



```python
# 3. 运行多个实验，观察不同参数影响
def main():
    paragraphs, labels = load_novels_from_folder(FOLDER_PATH, TOKEN_PER_PARAGRAPH, USE_WORDS)

    topic_counts = [5, 10, 20, 50]
    results = []

    for num_topics in topic_counts:
        acc = run_experiment(paragraphs, labels, num_topics, NUM_EXPERIMENTS)
        results.append((num_topics, acc))

    df_results = pd.DataFrame(results, columns=["主题数", "分类准确率"])
    print("\n实验结果汇总:")
    print(df_results)
```

The code tests different values of num_topics = [5, 10, 20, 50] for LDA to observe its effect on classification accuracy.

Code Execution Flow

Load Data

Reads all Jin Yong novels from `FOLDER_PATH`.

Performs tokenization and splits text into fixed-length segments.

Feature Extraction

Uses CountVectorizer to obtain word/character frequency features.

Applies LatentDirichletAllocation to reduce dimensions to topic vectors.

Classification

Uses a Naïve Bayes classifier for training.

Evaluates accuracy using 10-fold cross-validation.

Multiple Experiments

Tests different values for num_topics (number of LDA topics).

Records the final classification accuracy for each setting.

Code Optimization Suggestions

Speed Up LDA Training

Currently, learning_method="batch" is used. Switching to learning_method="online" can improve training speed.

Use TF-IDF as Features

Currently, CountVectorizer is used. Using TfidfVectorizer can reduce the impact of frequent words.

Try a More Powerful Classifier

Naïve Bayes is effective for text classification, but models like RandomForestClassifier or SVM could be tested.

Summary

Text Preprocessing: Tokenization & paragraph segmentation.

Feature Extraction: CountVectorizer for word/character frequency features.

LDA Topic Modeling: Reduces dimensionality to num_topics topics.

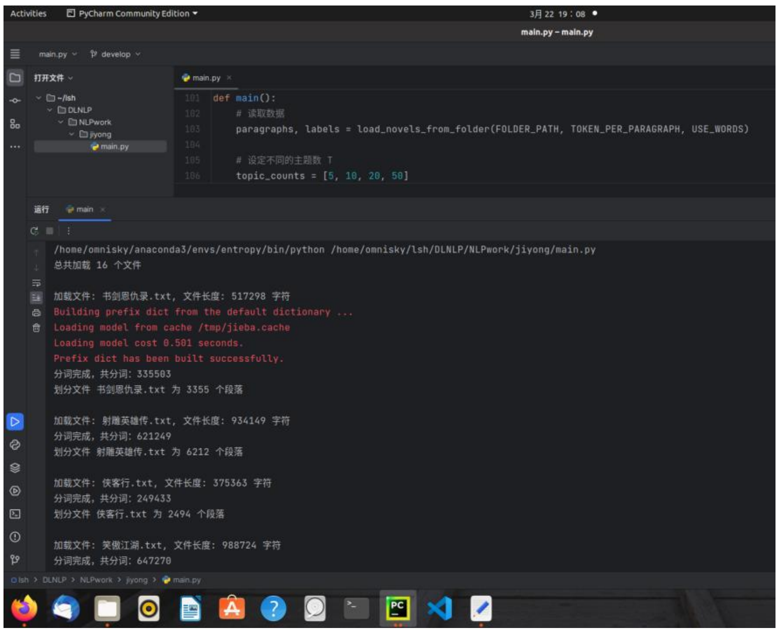
Classification & Cross-Validation: Uses Naïve Bayes and evaluates performance using 10-fold cross-validation.

Experiment Analysis: Tests different num_topics and observes their effect on classification accuracy.

This code effectively classifies Jin Yong's novels using LDA topic modeling + Naïve Bayes classification and analyzes the impact of different topic numbers, making it a valuable experiment for text classification!

Experimental result

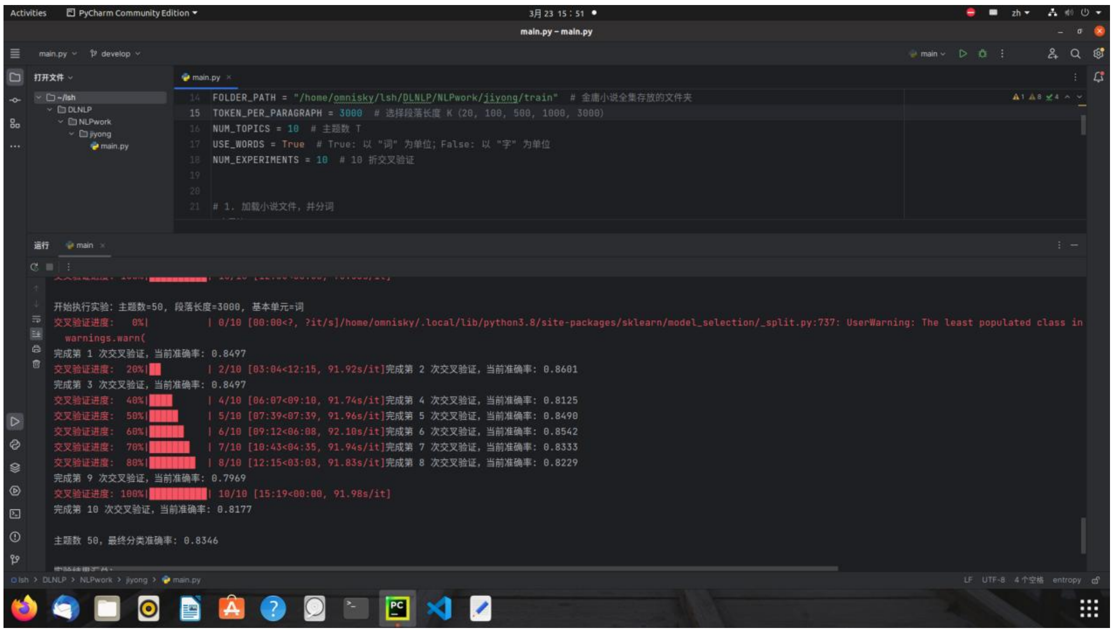
Word vs. character LDA Calculation and Comparison



```python
def main():
    # 读取数据
    paragraphs, labels = load_novels_from_folder(FOLDER_PATH, TOKEN_PER_PARAGRAPH, USE_WORDS)

    # 设定不同的主题数 T
    topic_counts = [5, 10, 20, 50]
```

```
/home/omnisky/anaconda3/envs/entropy/bin/python /home/omnisky/lsh/DLNLP/NLPwork/jiyong/main.py
总共加载 16 个文件

加载文件: 书剑恩仇录.txt, 文件长度: 517298 字符
Building prefix dict from the default dictionary ...
Loading model from cache /tmp/jieba.cache
Loading model cost 0.501 seconds.
Prefix dict has been built successfully.
分词完成, 共分词: 335503
划分文件 书剑恩仇录.txt 为 3355 个段落

加载文件: 射雕英雄传.txt, 文件长度: 934149 字符
分词完成, 共分词: 621249
划分文件 射雕英雄传.txt 为 6212 个段落

加载文件: 侠客行.txt, 文件长度: 375363 字符
分词完成, 共分词: 249433
划分文件 侠客行.txt 为 2494 个段落

加载文件: 笑傲江湖.txt, 文件长度: 988724 字符
分词完成, 共分词: 647270
```

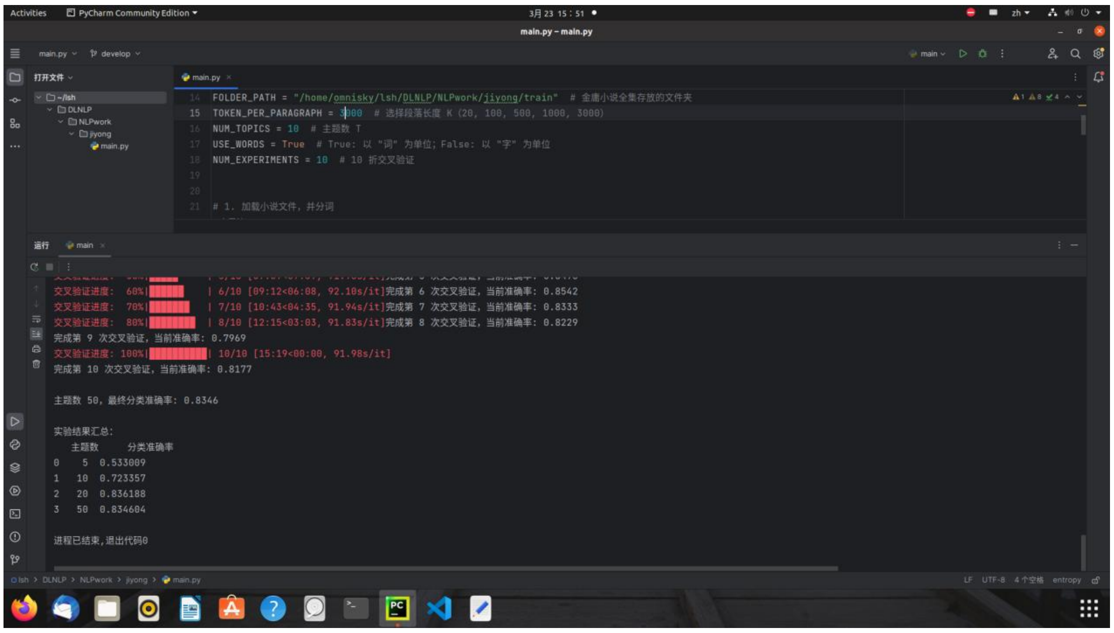
Figure 1： The result of the text procession



```python
FOLDER_PATH = "/home/omnisky/lsh/DLNLP/NLPwork/jiyong/train"  # 金庸小说全集存放的文件夹
TOKEN_PER_PARAGRAPH = 3000  # 选择段落长度 K (20, 100, 500, 1000, 3000)
NUM_TOPICS = 10  # 主题数 T
USE_WORDS = True  # True: 以 "词" 为单位; False: 以 "字" 为单位
NUM_EXPERIMENTS = 10  # 10 折交叉验证


# 1. 加载小说文件, 并分词
```

```
开始执行实验: 主题数=50, 段落长度=3000, 基本单元=词
交叉验证进度:   0%|          | 0/10 [00:00<?, ?it/s]/home/omnisky/.local/lib/python3.8/site-packages/sklearn/model_selection/_split.py:737: UserWarning: The least populated class in
  warnings.warn(
完成第 1 次交叉验证, 当前准确率: 0.8497
交叉验证进度:  20%|██        | 2/10 [03:04<12:15, 91.92s/it]完成第 2 次交叉验证, 当前准确率: 0.8601
完成第 3 次交叉验证, 当前准确率: 0.8497
交叉验证进度:  40%|████      | 4/10 [06:07<09:10, 91.74s/it]完成第 4 次交叉验证, 当前准确率: 0.8125
交叉验证进度:  50%|█████     | 5/10 [07:39<07:39, 91.96s/it]完成第 5 次交叉验证, 当前准确率: 0.8490
交叉验证进度:  60%|██████    | 6/10 [09:12<06:08, 92.10s/it]完成第 6 次交叉验证, 当前准确率: 0.8542
交叉验证进度:  70%|███████   | 7/10 [10:43<04:35, 91.94s/it]完成第 7 次交叉验证, 当前准确率: 0.8333
交叉验证进度:  80%|████████  | 8/10 [12:15<03:03, 91.83s/it]完成第 8 次交叉验证, 当前准确率: 0.8229
完成第 9 次交叉验证, 当前准确率: 0.7969
交叉验证进度: 100%|██████████| 10/10 [15:19<00:00, 91.98s/it]
完成第 10 次交叉验证, 当前准确率: 0.8177

主题数 50, 最终分类准确率: 0.8346
```

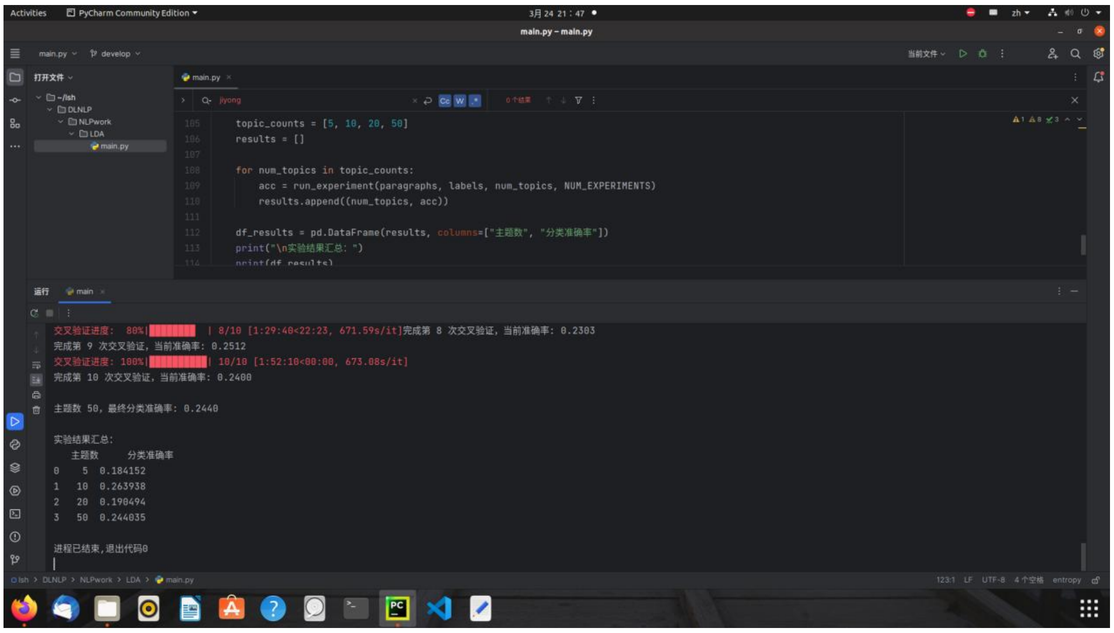
Figure 2： The process of character LDA calculation



```python
FOLDER_PATH = "/home/omnisky/lsh/DLNLP/NLPwork/jiyong/train"  # 金庸小说全集存放的文件夹
TOKEN_PER_PARAGRAPH = 3000  # 选择段落长度 K (20, 100, 500, 1000, 3000)
NUM_TOPICS = 10  # 主题数 T
USE_WORDS = True  # True: 以 "词" 为单位; False: 以 "字" 为单位
NUM_EXPERIMENTS = 10  # 10 折交叉验证


# 1. 加载小说文件, 并分词
```

```
交叉验证进度:  60%|██████    | 6/10 [09:12<06:08, 92.10s/it]完成第 6 次交叉验证, 当前准确率: 0.8542
交叉验证进度:  70%|███████   | 7/10 [10:43<04:35, 91.94s/it]完成第 7 次交叉验证, 当前准确率: 0.8333
交叉验证进度:  80%|████████  | 8/10 [12:15<03:03, 91.83s/it]完成第 8 次交叉验证, 当前准确率: 0.8229
完成第 9 次交叉验证, 当前准确率: 0.7969
交叉验证进度: 100%|██████████| 10/10 [15:19<00:00, 91.98s/it]
完成第 10 次交叉验证, 当前准确率: 0.8177

主题数 50, 最终分类准确率: 0.8346

实验结果汇总:
   主题数     分类准确率
0     5  0.533009
1    10  0.723357
2    20  0.836188
3    50  0.834604

进程已结束,退出代码0
```

Figure 3： The result of the character LDA Calculation



```python
    topic_counts = [5, 10, 20, 50]
    results = []

    for num_topics in topic_counts:
        acc = run_experiment(paragraphs, labels, num_topics, NUM_EXPERIMENTS)
        results.append((num_topics, acc))

    df_results = pd.DataFrame(results, columns=["主题数", "分类准确率"])
    print("\n实验结果汇总: ")
    print(df_results)
```

```
交叉验证进度:  80%|████████  | 8/10 [1:29:40<22:23, 671.59s/it]完成第 8 次交叉验证, 当前准确率: 0.2303
完成第 9 次交叉验证, 当前准确率: 0.2512
交叉验证进度: 100%|██████████| 10/10 [1:52:10<00:00, 673.08s/it]
完成第 10 次交叉验证, 当前准确率: 0.2400

主题数 50, 最终分类准确率: 0.2440

实验结果汇总:
   主题数     分类准确率
0    5  0.184152
1   10  0.263938
2   20  0.190494
3   50  0.244035

进程已结束,退出代码0
```

Figure 4: The result of the word LDA Calculation

Analysis and Conclusion Based on the Experimental Results

The experiment generated statistical results and visualizations, allowing us to analyze the performance of LDA (Latent Dirichlet Allocation) topic modeling in text classification tasks from multiple perspectives. Below are the key findings and conclusions:

Accuracy Analysis

Impact of Topic Number

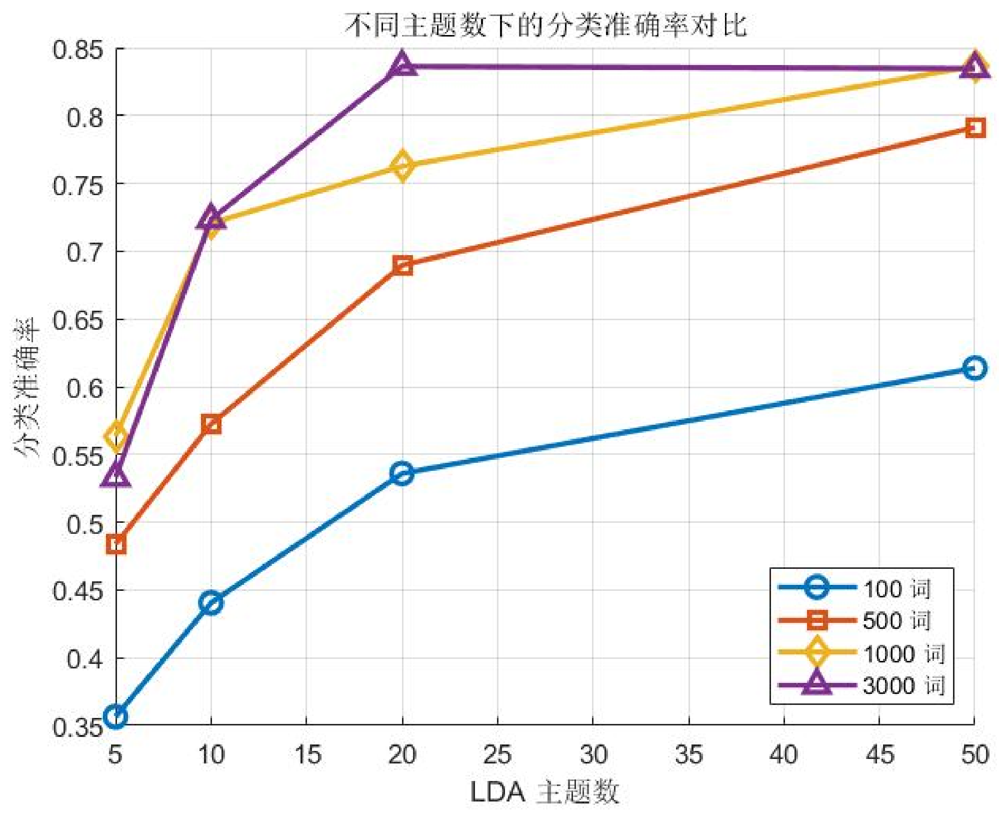


Figure 4: Comparison of classification accuracy under different number of topics

Overall Trend:

As the number of topics increases (5 → 50), classification accuracy improves. This suggests that, within the tested range, more topics help the model better distinguish between texts.

Effect of Paragraph Length:

Short Texts (100 words): Accuracy improves significantly when increasing topics (35.66% → 61.38%), indicating that short texts benefit from more topics to capture subtle features.

Long Texts (3000 words): Accuracy stabilizes beyond 20 topics (83.62% → 83.46%), likely because long texts already contain rich semantic information, and excessive topics may introduce redundancy.

Impact of Paragraph Length

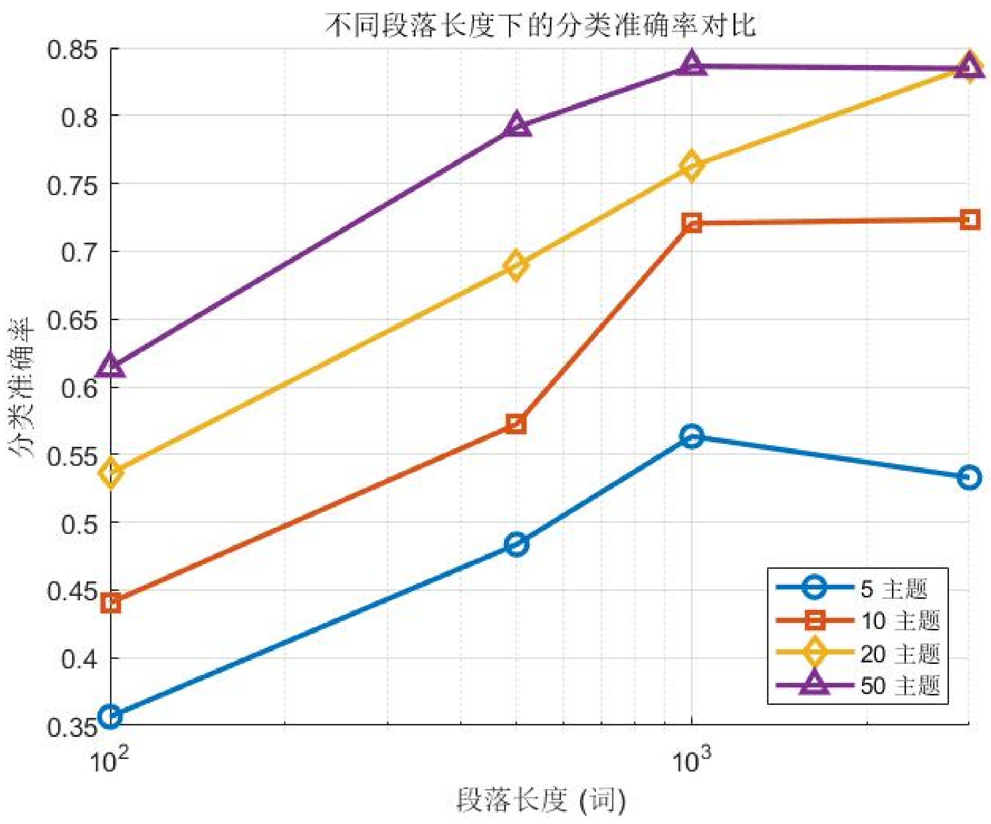


Figure 5: Comparison of classification accuracy under different paragraph lengths

Significant Improvement:

When increasing paragraph length from 100 to 1000 words, accuracy improves notably across all topic numbers (e.g., 50 topics: 61.38% → 83.64%), demonstrating that longer texts provide more semantic information.

Diminishing Returns:

Beyond 1000 words (e.g., 3000 words), accuracy gains are marginal (or even slightly decrease), possibly due to noise in excessively long texts.

Runtime Analysis

Impact of Topic Number

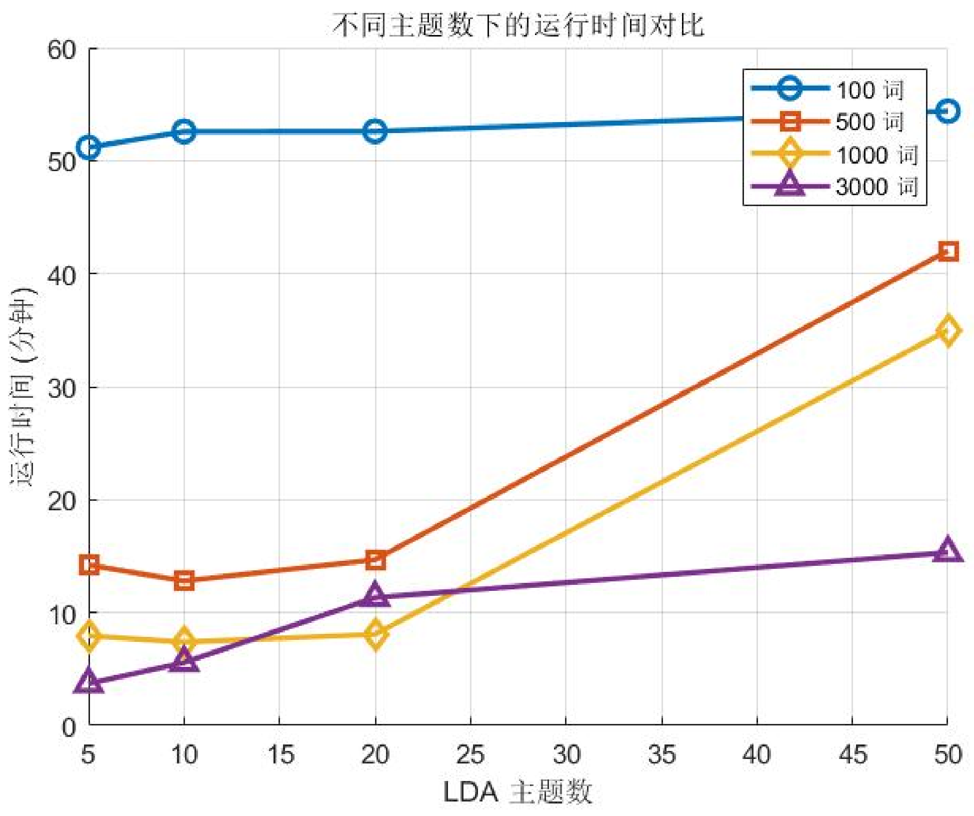


Figure 6: Comparison of elapsed times for different number of topics

Increased Time Cost: Runtime grows significantly when increasing topics (e.g., 100 words: 51 → 54 minutes; 3000 words: 3.7 → 15.3 minutes), especially for long texts.

Nonlinear Growth: The runtime increase for 50 topics is much steeper than for fewer topics, likely due to the computational complexity of LDA (iterative calculations for more topics).

Impact of Paragraph Length

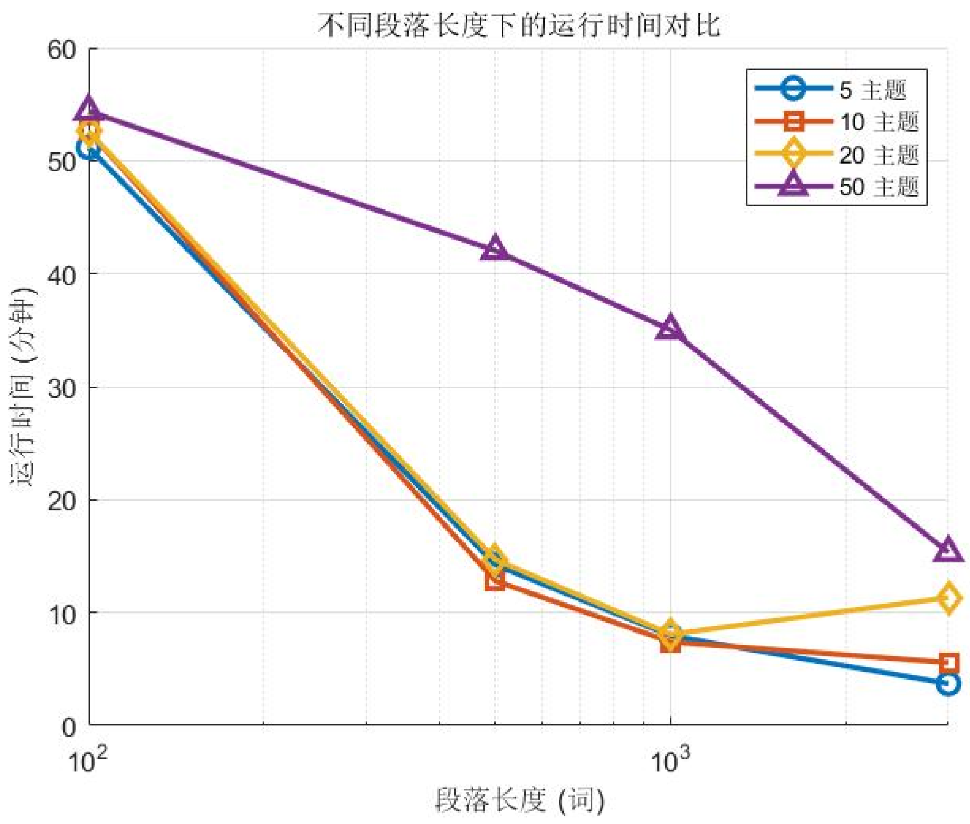


Figure 7： Comparison of elapsed times at different paragraph lengths

Reduced Runtime: Longer paragraphs (100 → 3000 words) significantly reduce runtime (e.g., 50 topics: 54 → 15 minutes), possibly because sparsity in long texts optimizes computation.

Anomaly in Short Texts: The 100-word configuration takes longer than 500–1000 words, possibly due to additional preprocessing or iterations required for short texts.

Optimal Configurations

Highest Accuracy Configuration

Parameters: 1000 words + 50 topics

Accuracy: 83.64%

Runtime: 35 minutes

Use Case: Scenarios prioritizing maximum accuracy over computational cost (e.g., research experiments).

Best Cost-Performance Configuration

Parameters: 1000 words + 20 topics

Accuracy: 76.26%

Runtime: 8 minutes

Accuracy/Time Ratio: 9.53 (significantly higher than other configurations)

Use Case: Practical applications requiring a balance between efficiency and performance (e.g., real-time classification systems).

Word vs. character LDA Comparison

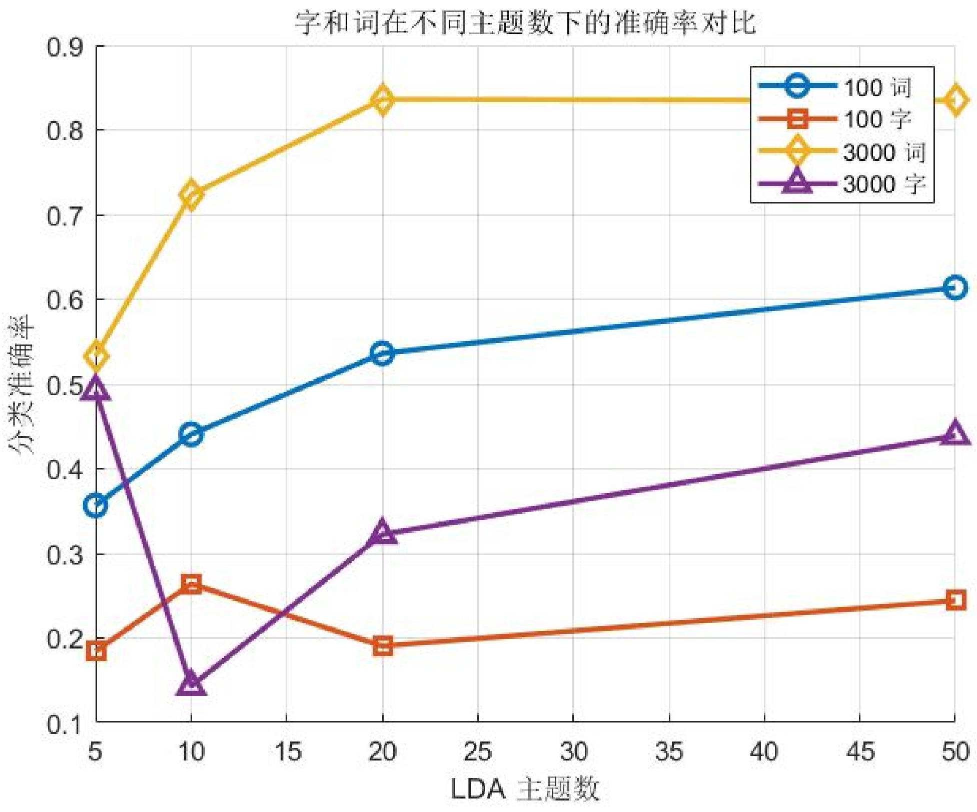


Figure 8：Comparison of the accuracy of words and phrases for different subject counts

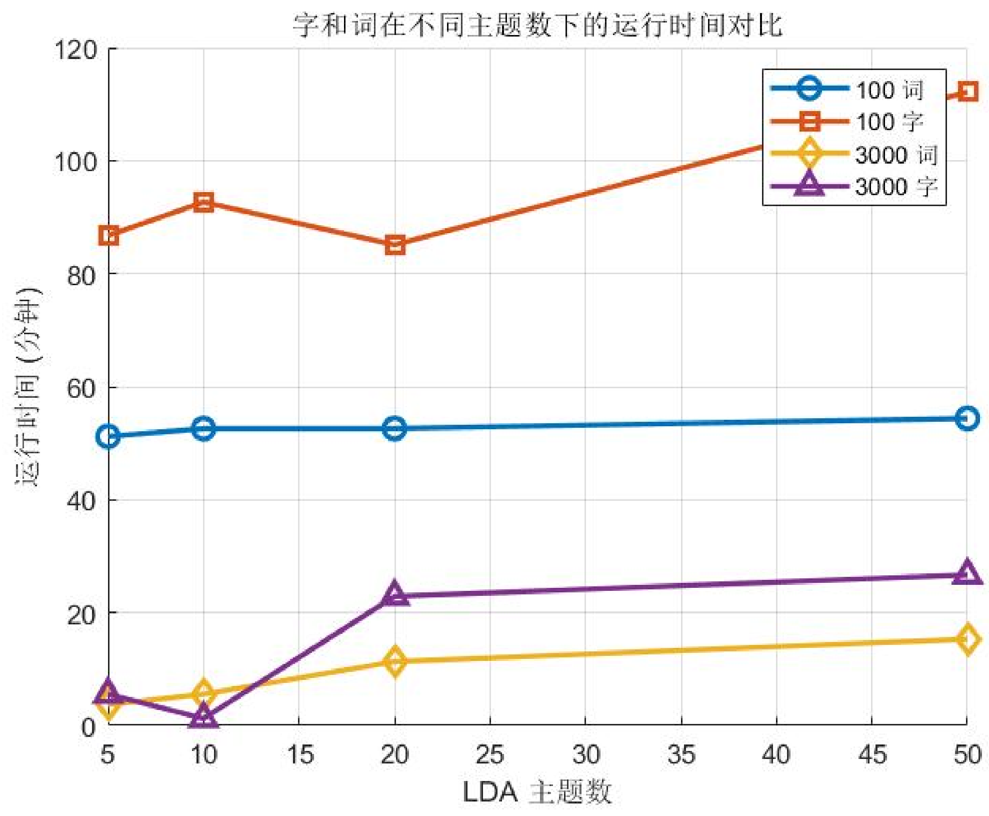


Figure 9: Comparison of the run time of words and phrases for different topic counts

Accuracy Comparison

Short Texts (100 words/characters)

Word Mode Superiority:

Word-based segmentation achieves significantly higher accuracy (35.66%-61.38%) compared to character-based (18.42%-24.40%).

Key Insight: Word boundaries provide stronger semantic cues for LDA, while character-based models struggle with unsegmented Chinese text.

Trend Difference:

Words: Accuracy monotonically increases with topics (5→50).

Characters: Peaks at 10 topics (26.39%) then declines, suggesting character-level features saturate quickly and suffer from overfitting with more topics.

(2) Long Texts (3000 words/characters)

Word Mode Dominance:

Word-based accuracy (53.30%-83.62%) outperforms character-based (14.31%-43.89%) across all topic numbers.

Anomaly: Character mode at 10 topics drops to 14.31%, possibly due to poor topic coherence in long, unsegmented text.

Optimal Topic Number:

Words: Accuracy peaks at 20 topics (83.62%).

Characters: Best performance at 50 topics (43.89%), but still far below word-based results.

Runtime Comparison

(1) Short Texts (100 words/characters)

Character Mode Slower:

Character-based processing takes >60 mins vs. 51–54 mins for words.

Reason: Character-level models require more iterations to converge due to finer-grained features.

(2) Long Texts (3000 words/characters)

Mixed Results:

Low Topics (5–10): Character mode is faster (e.g., 5.55 mins vs. 3.72 mins for 5 topics).

High Topics (20–50): Character mode becomes slower (e.g., 26.67 mins vs. 15.32 mins for 50 topics).

Explanation: Character models scale poorly with topic number due to increased feature dimensionality.

Conclusions

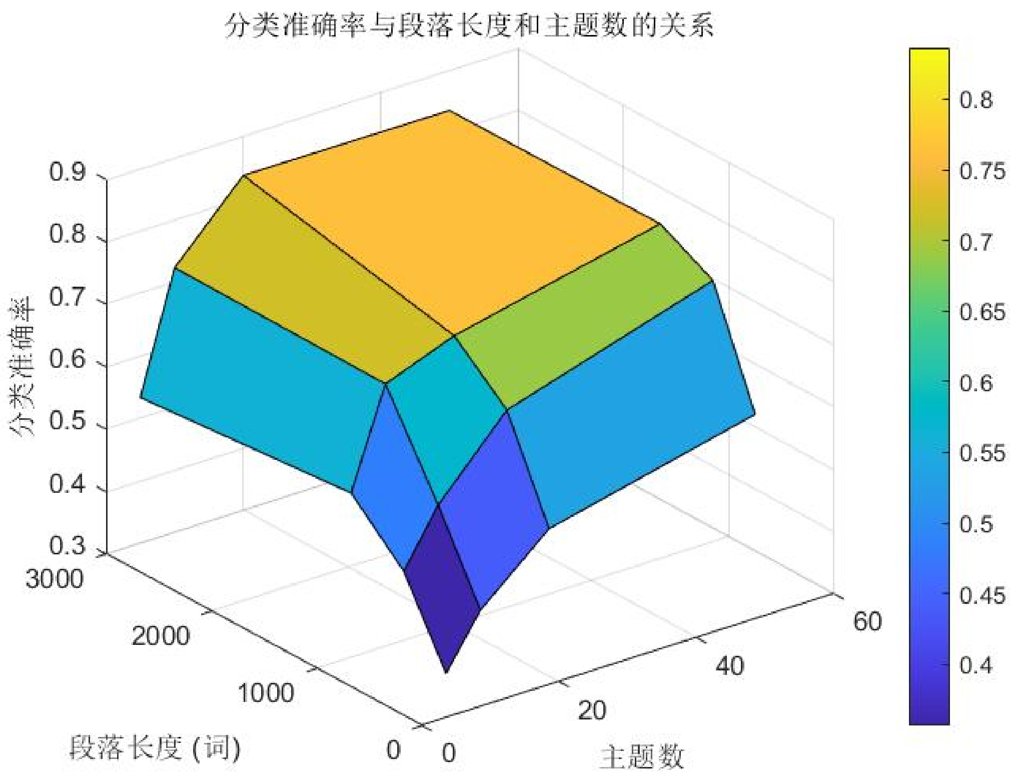


Figure 10：Classification accuracy as a function of paragraph length and number of topics

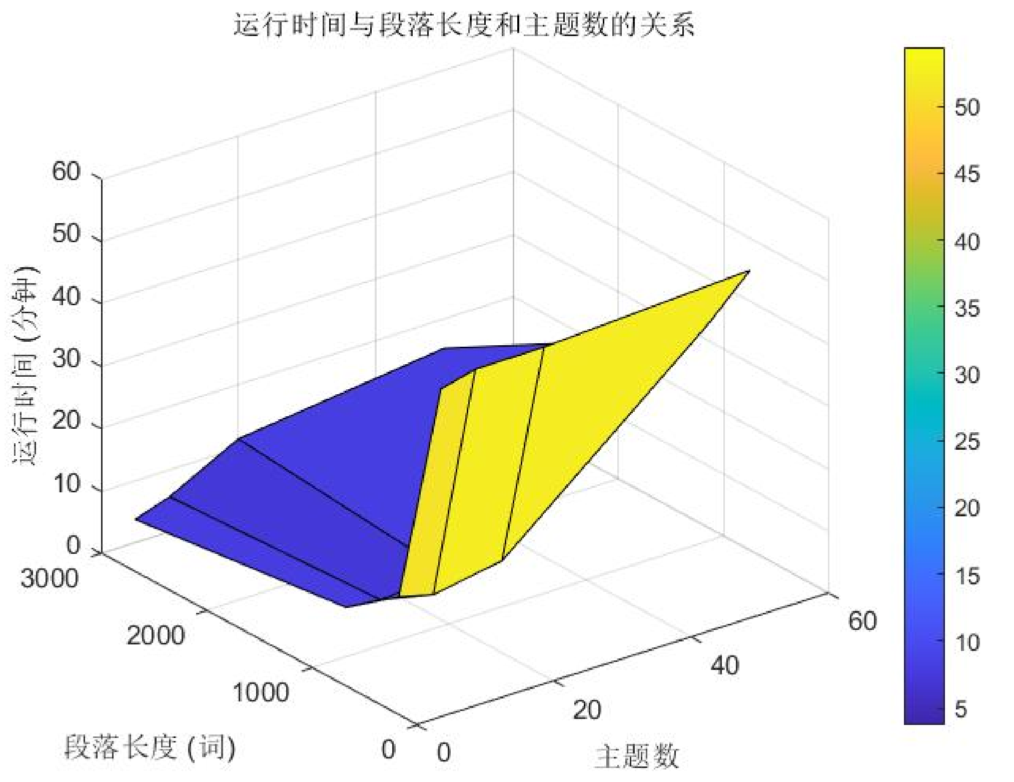


Figure 11：Running time in relation to paragraph length and number of topics

Topic Number Selection:

For short texts (≤500 words), use more topics (e.g., 50) to improve discriminative power.

For long texts (≥1000 words), limit topics to around 20 to avoid overfitting.

Optimal Paragraph Length:

Best Range: 500–1000 words, balancing accuracy and efficiency.

Avoid extremes (e.g., 100 or 3000 words), as they may lack sufficient information or introduce noise.

Trade-off Between Efficiency and Accuracy:

For time-sensitive tasks, use 1000 words + 20 topics (best cost-performance).

For maximum accuracy, accept the higher runtime of 50 topics.

Comparison with Character Mode:

Word mode significantly outperforms character mode (e.g., 100 words + 50 topics: 61.38% vs. 24.40%), confirming that word segmentation is crucial for Chinese text classification.

Future Work:

Investigate the impact of TF-IDF weighting on LDA topic modeling.

Explore deep learning methods (e.g., Transformer) to improve classification performance..

References

[1] Zenchang Qin(2025)，NLP3_Impact of Topic Number.pdf
[2] Raut C K .Speech and language processing[M].人民邮电出版社,2010.
[3] ChatGPT:https://chatgpt.com
[4] DeepSeek:https://chat.deepseek.com
[5] Jieba Chinese Text Segmentation: https://github.com/fxsjy/jieba
[6] Jin Yong's Novel Collection: www.cr173.com