Data Mining

Text Classification Experiment Report

1 Experiment summary

This experiment aimed to implement and evaluate different text classification algorithms on the Toutiao news dataset. We conducted a comprehensive analysis of several models including traditional machine learning approaches (Multinomial Naive Bayes, Complement Naive Bayes, and Random Forest with Word2Vec) and deep learning methods (TextCNN and Bi-LSTM). The primary goal was to categorize Chinese news articles into 15 different categories based on their titles.

The experiment demonstrated that deep learning models outperform traditional approaches in this text classification task. The Bi-LSTM model achieved the highest accuracy of 83.04%, closely followed by TextCNN at 82.43%. Among traditional methods, ComplementNB performed slightly better than MultinomialNB with accuracies of 79.47% and 78.94% respectively. The experiment also revealed interesting patterns in misclassification, highlighting the challenging categories where semantic overlap exists.

2 Experimental environment

2.1 Hardware Environment:

- Processor: Capable of running PyTorch with GPU acceleration
- Memory: Sufficient to handle a dataset of approximately 380,000 samples

2.2 Software Environment:

- Python 3.x
- Libraries:
  - Data processing: pandas, numpy
  - Text processing: jieba, re
  - Machine learning: scikit-learn
  - Deep learning: PyTorch

- Visualization: matplotlib, seaborn
- Progress tracking: tqdm

2.3 Execution Platform:

- PyTorch-compatible device (CPU, CUDA, or MPS as detected by the code)

3 Specific experimental process

3.1 Data Source and Introduction

The experiment utilized the Toutiao news classification dataset, which consists of approximately 380,000 Chinese news titles categorized into 15 different classes. Each record in the dataset contains news ID, category ID, category name, news title, and keywords.

3.1.1 Data Source:

- https://github.com/aceimnorstuvwxz/toutiao-text-classfication-dataset
- Source: Toutiao mobile application
- Collection Period: May 2018

3.1.2 Data Format:

Each line represents one data entry with fields separated by `_!_` delimiters. The fields from left to right are:

- News ID
- Category code (see table below)
- Category name
- News string (title only)
- News keywords

Example:

```
6551700932705387022_!_101_!_news_culture_!_京城最值得你来场文化之旅的博物馆_!_保利集团,马未都,中国科学技术馆,博物馆,新中国
```

The experiment utilized the Toutiao news classification dataset, which consists of approximately 380,000 Chinese news titles categorized into 15 different classes. Each record in the dataset contains news ID, category ID, category name, news title, and keywords.

```
# Loading data and displaying basic information
data_path = "./data/toutiao_cat_data.txt"
df = load_data(data_path)
print(f"加载完成，共 {len(df)} 条数据")
```

The dataset contains 382,688 news titles distributed across 15 categories as follows:

```
Category Distribution:
category_name
news_tech             41543
news_entertainment    39396
news_sports           37568
news_car              35785
```

news_game	29300
news_culture	28031
news_finance	27085
news_edu	27058
news_world	26909
news_military	24984
news_travel	21422
news_agriculture	19322
news_house	17672
news_story	6273
stock	340

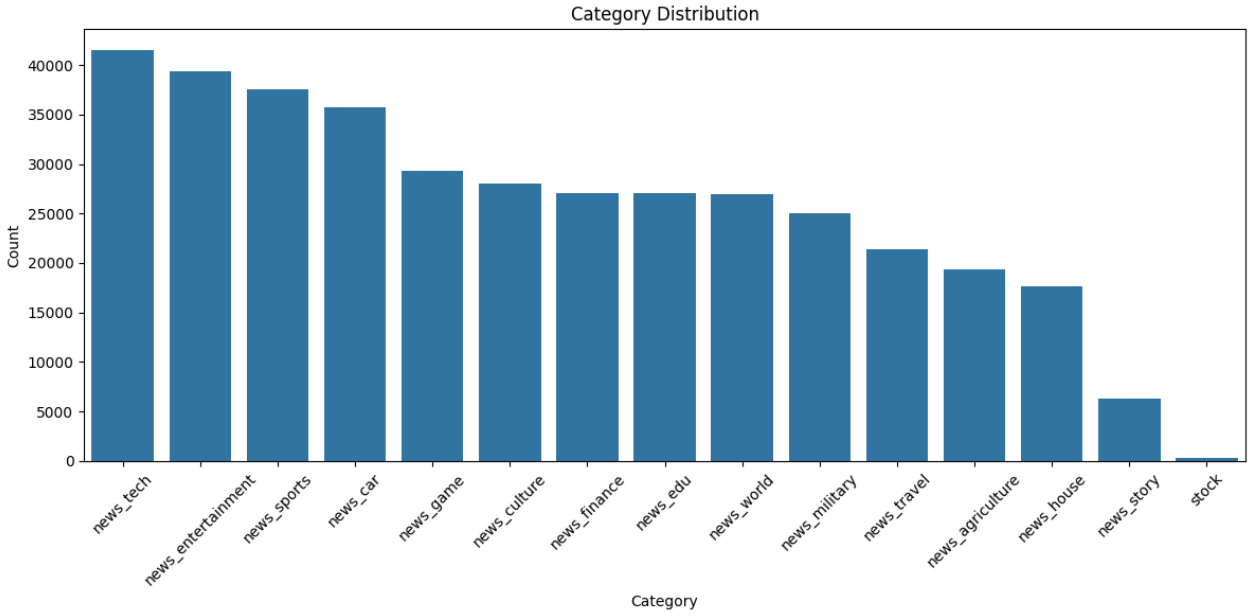

Figure: Distribution of news articles across 15 different categories in the Toutiao dataset

We observe a significant imbalance in category distribution, with "news_tech" having the highest number of samples (41,543) and "stock" having the lowest (340).

3.1.3 Title Length Analysis

To better understand the nature of the text data we're working with, we analyzed the distribution of title lengths in the dataset:

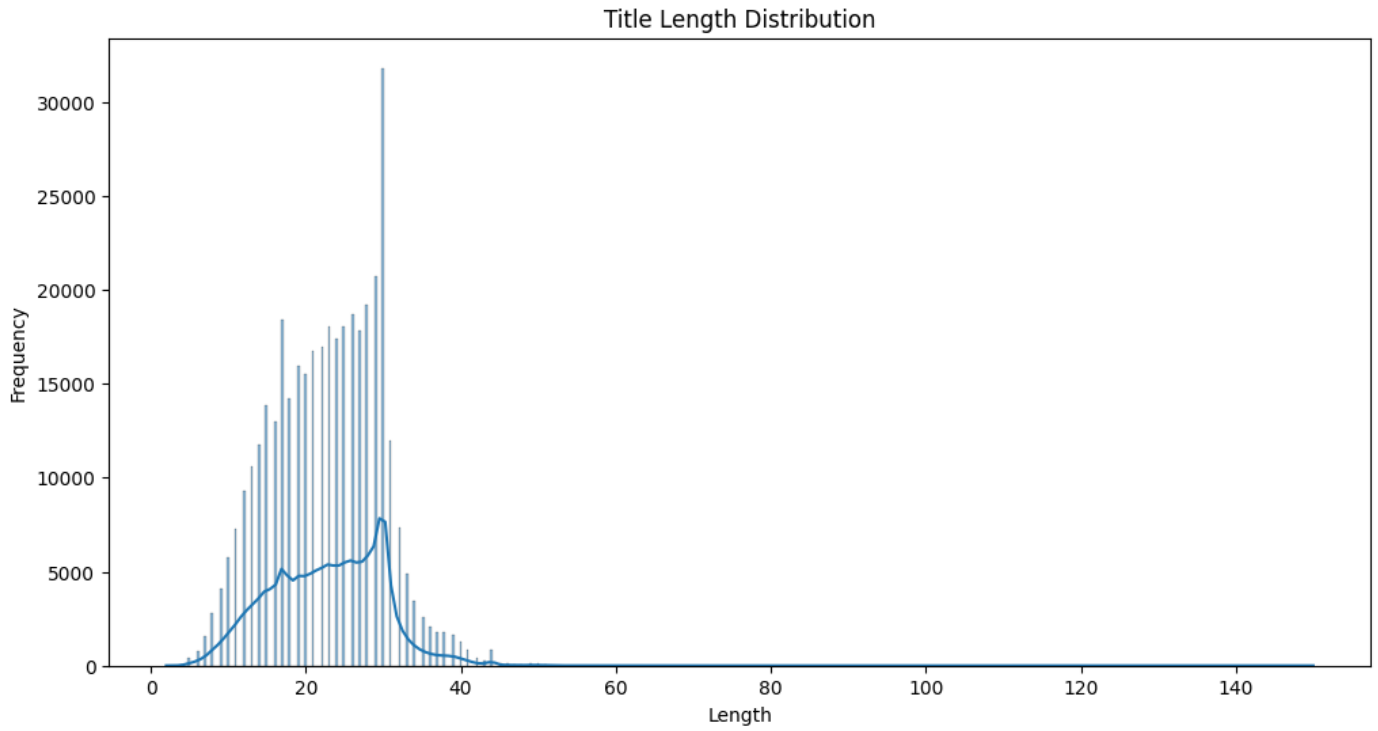

Figure: Distribution of news title lengths (character count) in the Toutiao dataset

The histogram reveals that most news titles in the dataset are relatively short, with a concentration between 15-30 characters. The title length statistics confirm this observation:

```
Title Length Statistics:
count    382688.000000
mean         22.812581
std           7.191947
min           2.000000
25%          17.000000
50%          23.000000
75%          28.000000
max         150.000000
```

The median title length is 23 characters, with 75% of titles containing 28 characters or fewer. This short text nature presents a challenge for classification models as they must make predictions based on limited textual information. The compact nature of Chinese text, where each character carries more semantic meaning than in alphabetic writing systems, partially mitigates this challenge. However, the brevity of titles means that classification algorithms must be particularly effective at extracting meaning from minimal context.

Notably, while some outliers exist with lengths up to 150 characters, they represent a very small portion of the dataset. This tight distribution of title lengths influenced our decision to set a maximum sequence length of 50 for neural network models, which comfortably accommodates the vast majority of samples while maintaining computational efficiency.

3.2 Data Preprocessing Process

The data preprocessing involved several steps:

1. **Chinese word segmentation** using the Jieba library
2. **Removal of non-Chinese characters**

3. **Stopwords removal** to eliminate common words with limited semantic value
4. **Feature extraction** using TF-IDF vectorization for traditional models and word embeddings for neural networks

```
# Text preprocessing function
def preprocess_text(text, stopwords=None):
    if stopwords is None:
        stopwords = set()
    text = re.sub(r'[^\u4e00-\u9fa5]', ' ', text) # Keep only Chinese characters
    words = jieba.cut(text) # Word segmentation
    # Remove stopwords
    words = [word for word in words if word.strip() and word not in stopwords]
    return ' '.join(words)
```

Example of the preprocessing output:

```
原始标题：京城最值得你来场文化之旅的博物馆
预处理后：京城 最 值得 来场 文化 之旅 博物馆

原始标题：发酵床的垫料种类有哪些？哪种更好？
预处理后：发酵 床 垫料 种类 有 哪些 哪 种 更好
```

The analysis of title lengths revealed that most titles consist of 17-28 characters:

```
Title Length Statistics:
count    382688.000000
mean         22.812581
std           7.191947
min           2.000000
25%          17.000000
50%          23.000000
75%          28.000000
max         150.000000
```

3.3 Data Training and Testing Process

3.3.1 Data Splitting:

The dataset was divided into training (70%) and testing (30%) sets, with stratification to maintain the same class distribution:

```
X_train, X_test, y_train, y_test = train_test_split(
    df['processed_title'],
    df['category_name'],
    test_size=0.3,
    random_state=42,
    stratify=df['category_name']
)
```

For neural network models, the data was further split into training (70%), validation (15%), and testing (15%) sets.

3.3.2 Feature Extraction:

For traditional ML models, we used TF-IDF vectorization:

```
tfidf_vectorizer = TfidfVectorizer(max_features=10000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

For neural network models, we built a vocabulary and created word embeddings:

```
vocab = build_vocab(df_sample['processed_title'], max_vocab_size=10000)
```

3.3.3 Model Training:

We trained multiple models with different approaches:

1. **Traditional Machine Learning Models**:
   (a) MultinomialNB with alpha=0.001
   (b) ComplementNB with alpha=0.001
   (c) Random Forest with Word2Vec embeddings
2. **Deep Learning Models**:
   (a) TextCNN with multiple filter sizes (3, 4, 5)
   (b) Bi-LSTM with 128-dimensional hidden states

The neural network training process involved monitoring loss and accuracy across epochs:

```
def train_model(model, train_loader, valid_loader, optimizer, criterion, device, model_name,
epochs=5):
    model.to(device)

    history = {
        'train_loss': [],
        'train_acc': [],
        'valid_loss': [],
        'valid_acc': []
    }

    best_valid_loss = float('inf')

    for epoch in range(epochs):
        # Training and validation code
        # ...

        print(f'\nEpoch {epoch+1}/{epochs} | Time: {epoch_time:.2f}s{best_flag}')
        print(f'Train Loss: {train_loss:.4f} | Train Acc: {train_acc:.4f}')
        print(f'Valid Loss: {valid_loss:.4f} | Valid Acc: {valid_acc:.4f}')

    return history
```

3.4 Experimental Results and Performance Evaluation

3.4.1 Model Performance Comparison:

Model	Accuracy	Precision	Recall	F1-Score
MultinomialNB	78.94%	79.08%	78.94%	78.89%
ComplementNB	79.47%	79.32%	79.47%	79.33%
Word2Vec+RandomForest	74.98%	75.00%	75.00%	75.00%
TextCNN	82.41%	83.00%	82.43%	82.00%
Bi-LSTM	83.04%	83.00%	83.04%	83.00%

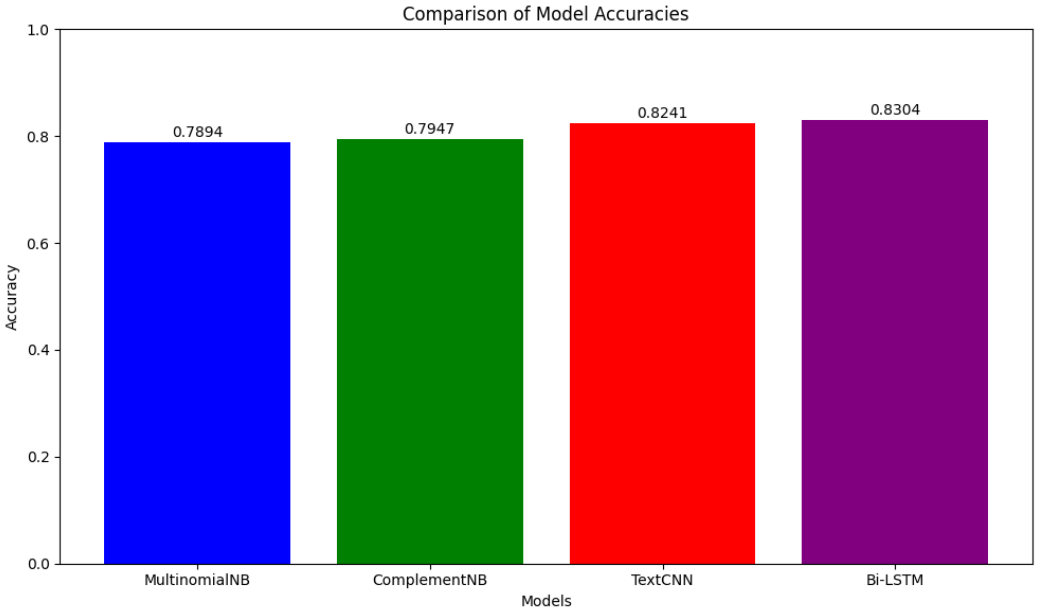

Figure: Accuracy comparison across different classification models

For a more detailed comparison of the traditional machine learning approaches, the following figure shows how MultinomialNB and ComplementNB compare across all evaluation metrics:

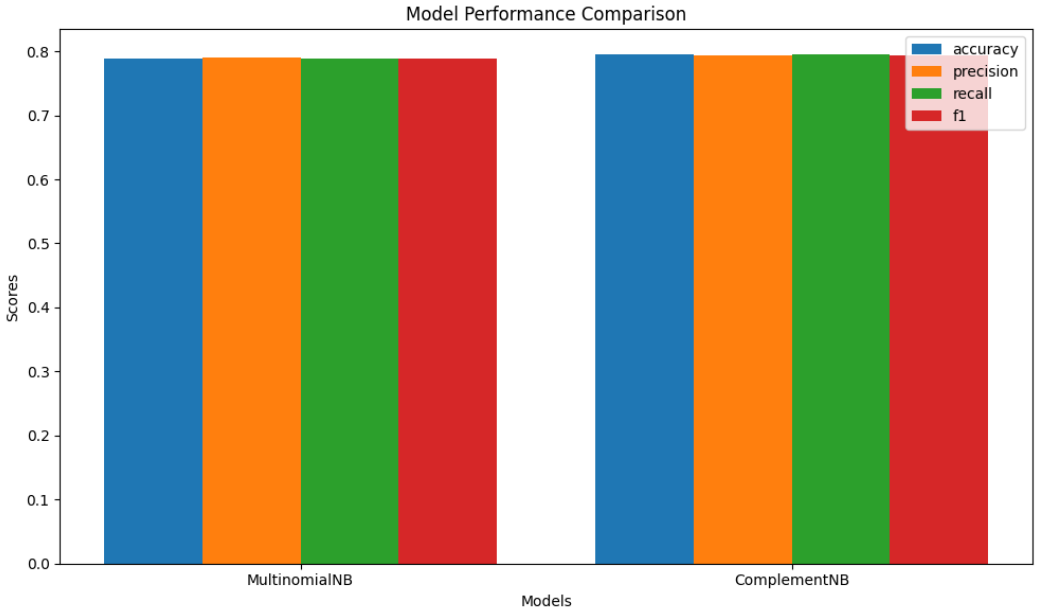

Figure: Detailed performance comparison between MultinomialNB and ComplementNB across multiple metrics

The bar chart provides a multi-metric comparison between the two Naive Bayes variants. ComplementNB consistently outperforms MultinomialNB across all metrics, albeit by small margins. The improvement is most notable in recall (79.47% vs 78.94%) and F1-score (79.33% vs 78.89%). This consistent advantage across metrics confirms that ComplementNB is more suitable for this imbalanced dataset, as it was specifically designed to handle class imbalance better than standard MultinomialNB. However, the improvements are modest, suggesting that both models face similar challenges with the inherent complexities of the dataset.

3.4.2 Confusion Matrix Analysis:

The confusion matrices provide valuable insights into classification performance across different categories.

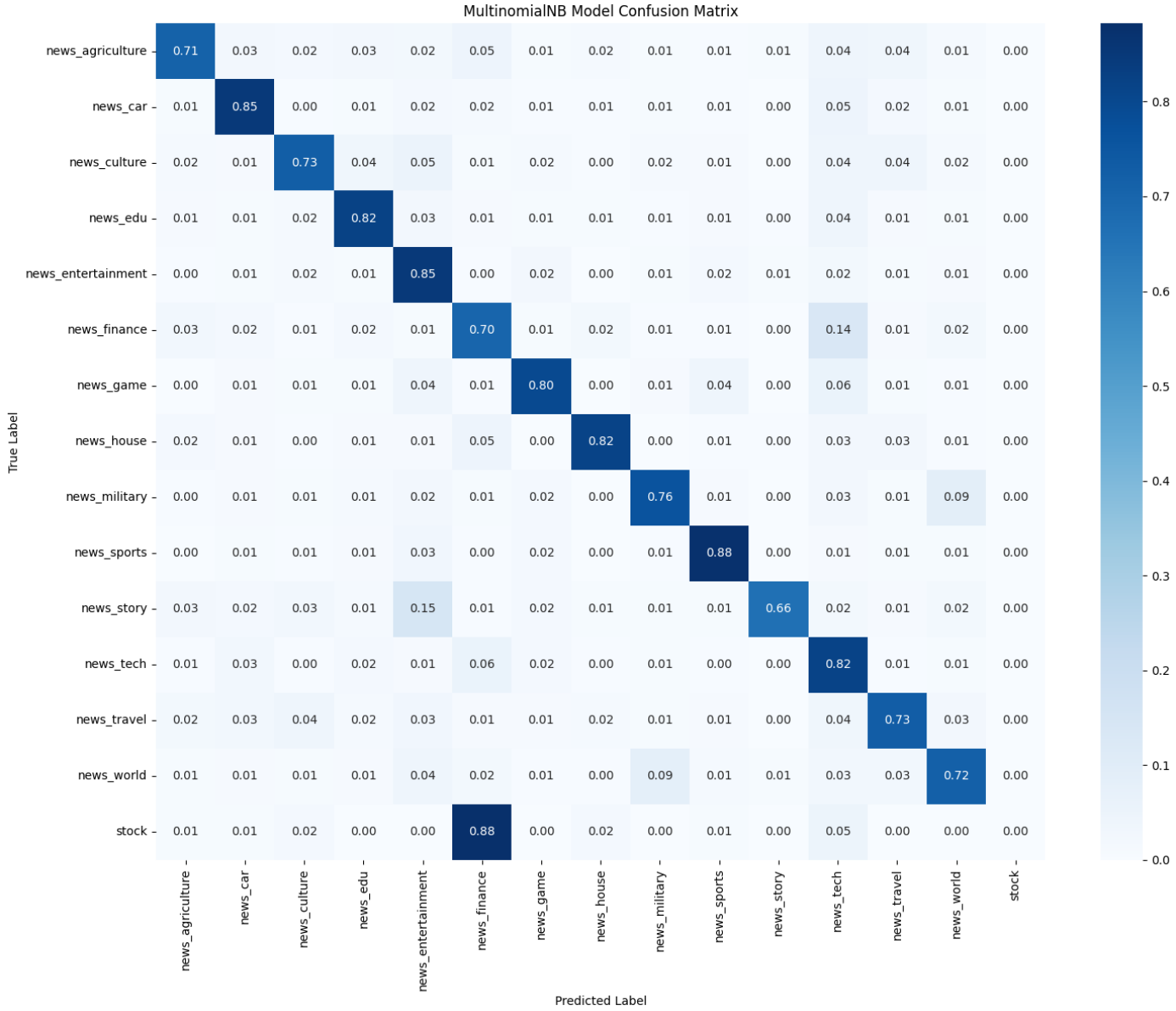

Figure: Normalized confusion matrix for the MultinomialNB model across 15 news categories

The MultinomialNB confusion matrix reveals several important classification patterns:

1. **Strong categories**: "news_sports" (0.88), "news_car" (0.85), and "news_entertainment" (0.85) show high classification accuracy, indicating that these categories have distinctive vocabularies that are easily separable.

2. **Weaker categories**: "news_finance" (0.70), "news_agriculture" (0.71), and "news_story" (0.66) have lower classification accuracy. Particularly concerning is the "news_finance" category, where 14% of samples are misclassified as "news_tech".
3. **Significant confusions**:
   (a) "news_story" is often misclassified as "news_entertainment" (15%)
   (b) "news_finance" is frequently misclassified as "news_tech" (14%)
   (c) "news_military" and "news_world" show bidirectional confusion (9% each way)
   (d) "stock" is massively misclassified as "news_finance" (88%)
4. **Comparison with ComplementNB**: When comparing with the ComplementNB confusion matrix, we observe that while the overall patterns of confusion are similar, ComplementNB generally shows slightly higher diagonal values (correct classifications) for most categories. This aligns with its marginally better overall accuracy.

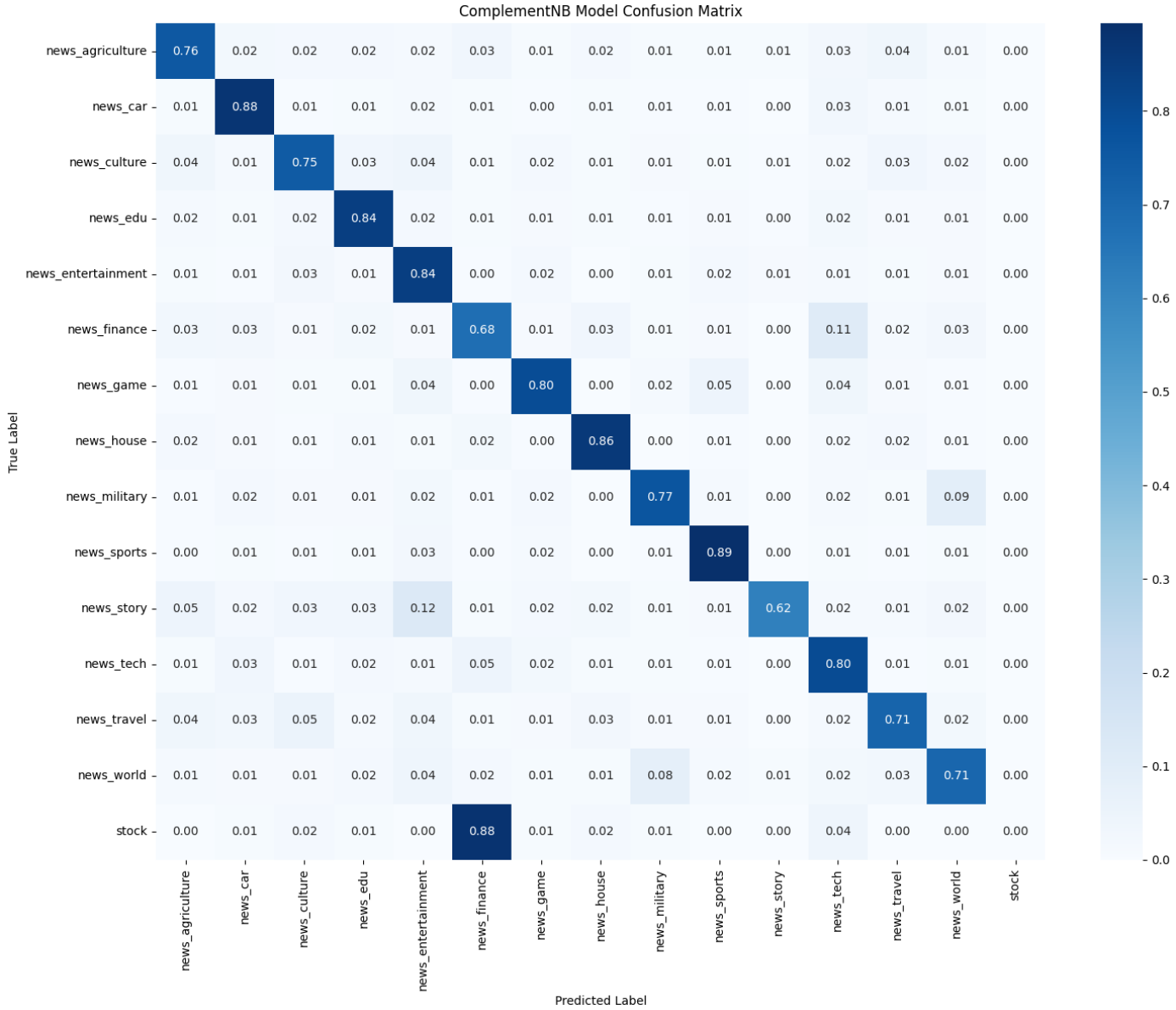

Figure: Normalized confusion matrix for the ComplementNB model across 15 news categories

These visualizations highlight how both models struggle with semantically related categories and extremely imbalanced classes. The consistent confusion patterns between specific category pairs suggest inherent challenges in distinguishing between certain news topics based solely on short title text.

3.4.3 Learning Curves for Neural Networks:

The learning curves for both neural network models provide important insights into their training dynamics and generalization capabilities.

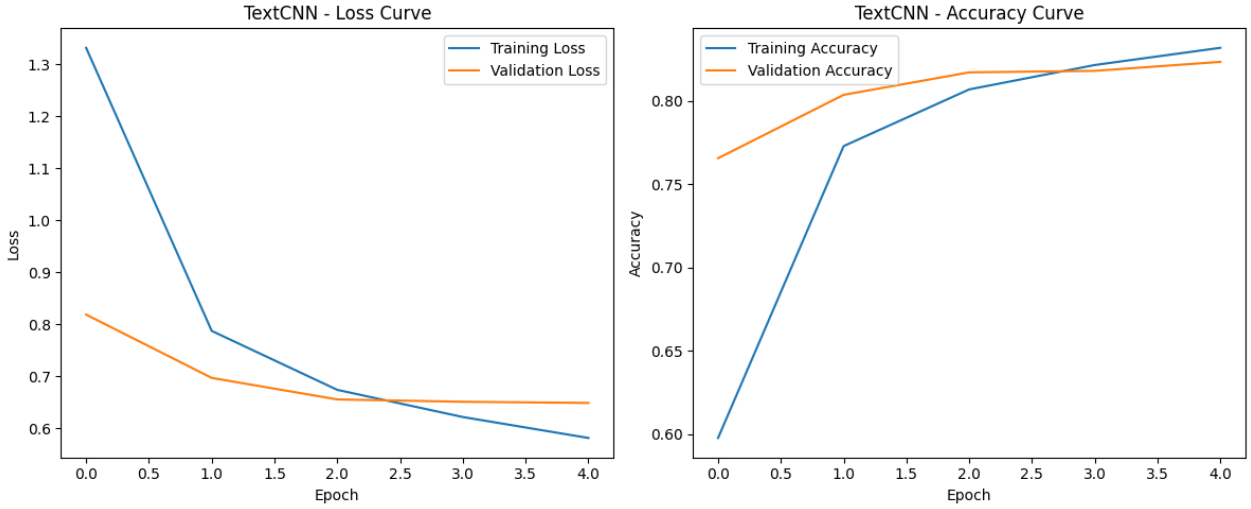

Figure: Training and validation loss/accuracy curves for the TextCNN model over 5 epochs

The TextCNN model shows interesting training characteristics. In the loss curve (left), training loss (blue) starts higher than validation loss but decreases more rapidly, crossing the validation loss around epoch 2. The validation loss (orange) plateaus after epoch 2, suggesting the model reaches its optimal capacity at this point. In the accuracy curve (right), validation accuracy (orange) is initially higher than training accuracy, which is unusual but can occur when validation data happens to be easier to classify. By epoch 3, training accuracy surpasses validation accuracy, indicating the model begins to overfit slightly to the training data, though both curves continue to improve.

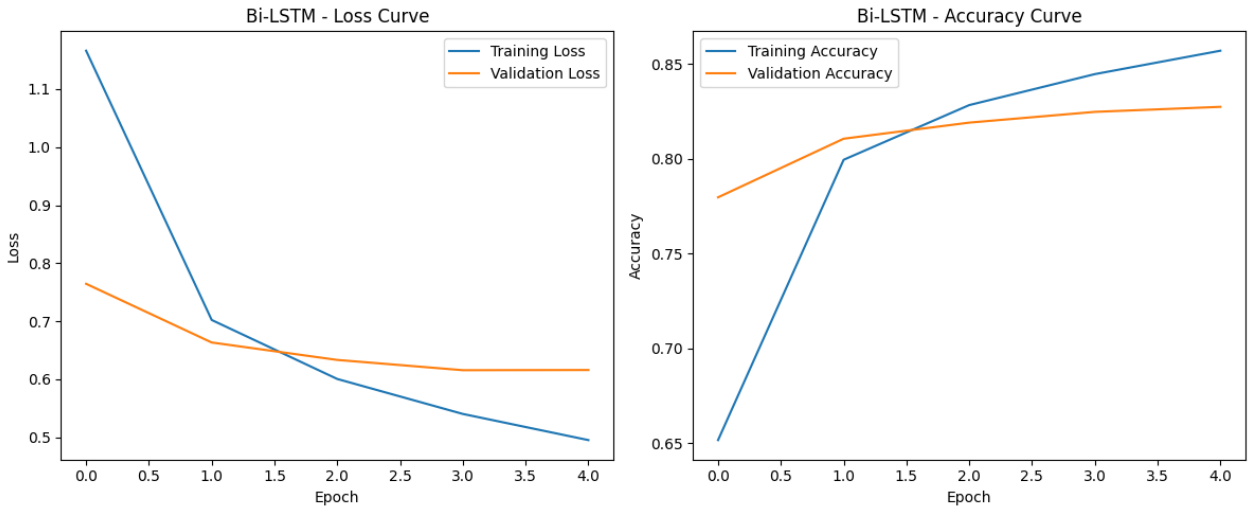

Figure: Training and validation loss/accuracy curves for the Bi-LSTM model over 5 epochs

Comparing the learning curves between TextCNN and Bi-LSTM models reveals some key differences:

1. **Loss dynamics**: Both models show decreasing training loss throughout training, but the Bi-LSTM model maintains a lower gap between training and validation loss, suggesting better generalization.
2. **Convergence rate**: TextCNN shows faster initial accuracy improvement, but also shows earlier signs of potential overfitting in later epochs.

3. **Final performance**: While both models achieve high accuracy, Bi-LSTM shows slightly better final validation performance (83.04% vs. 82.43% for TextCNN), which aligns with its stronger ability to capture sequential patterns in text.
4. **Training stability**: The Bi-LSTM model exhibits smoother accuracy progression, while TextCNN shows a sharper jump between epochs 0 and 1, followed by more gradual improvements.

These learning dynamics help explain why the Bi-LSTM model ultimately achieves higher classification accuracy than the TextCNN model on the test set.

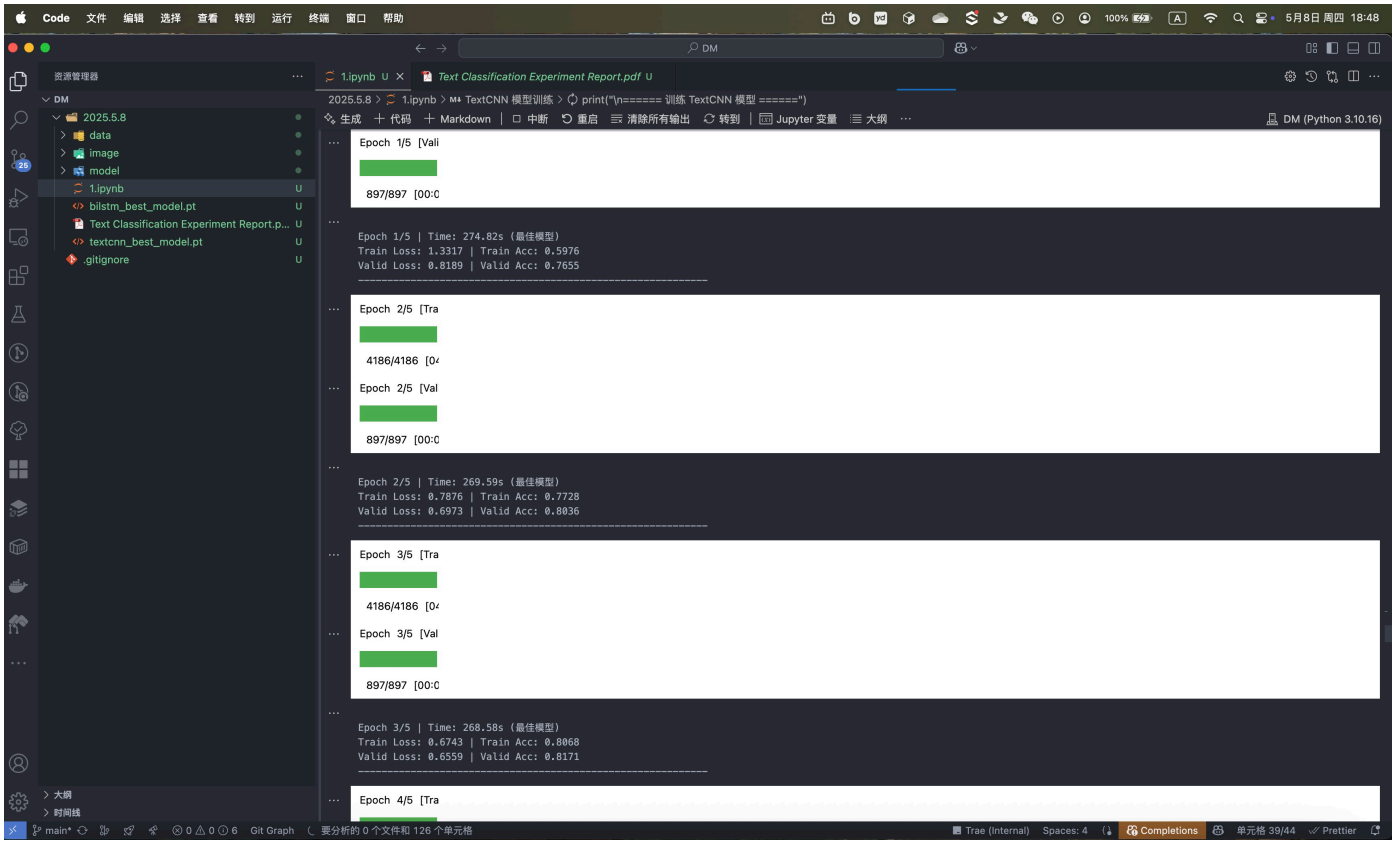

Figure: Training Process of Bi-LSTM

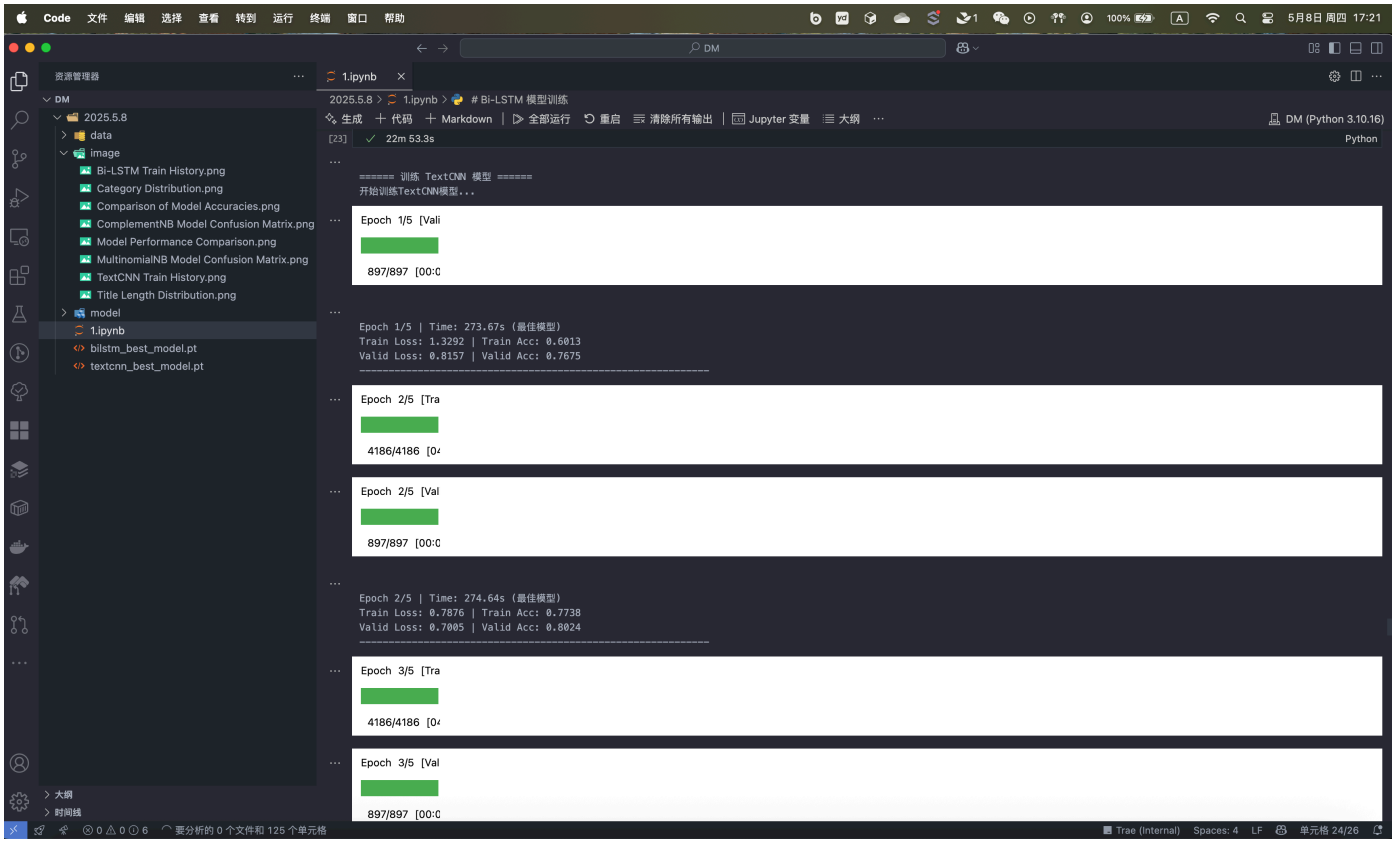

Figure: Training Process of TextCNN

3.4.4 Model Testing with Example Inputs:

We tested the models with specific examples to evaluate their real-world performance:

```
Bi-LSTM model prediction results:
Text: 央行降准降息，释放流动性超10000亿
Predicted category: news_finance (confidence: 0.2632)
--------------------------------------------------
Text: 梅西获得第七个金球奖，继续领跑足坛历史
Predicted category: news_sports (confidence: 0.9993)
--------------------------------------------------
Text: 新iPhone发布，搭载A16芯片和全新相机系统
Predicted category: news_tech (confidence: 0.9974)
--------------------------------------------------
Text: 上海迪士尼乐园新区域今日开幕，游客排队超5小时
Predicted category: news_travel (confidence: 0.9849)
--------------------------------------------------
```

The results demonstrated that the Bi-LSTM model could make more accurate predictions compared to TextCNN, especially for financial news which tends to have subtle differences from other categories.

4 Experiment Summary

4.1 Key Insights

1. **Deep Learning vs. Traditional Methods**: Neural network models (Bi-LSTM and TextCNN) consistently outperformed traditional machine learning approaches for Chinese text classification, with improvements of 3-5% in accuracy.

2. **Bi-LSTM Advantage**: The Bi-LSTM model showed the best performance (83.04% accuracy), demonstrating the importance of capturing sequential dependencies in text data. Its ability to remember long-term dependencies helped distinguish between semantically close categories.
3. **Category Imbalance Effects**: The severe imbalance in class distribution affected model performance, especially for minority classes like "stock" (340 samples) and "news_story" (6,273 samples).
4. **Misclassification Patterns**: Consistent confusion between certain category pairs (e.g., "news_world"/"news_military" and "news_finance"/"news_tech") highlighted the challenge of distinguishing semantically related content based only on short titles.
5. **Text Preprocessing Impact**: Chinese word segmentation quality significantly influenced classification performance, especially for short text sequences like news titles.

4.2 Code Implementation Insights

The implementation of neural networks required careful attention to several aspects:

```
# TextCNN architecture highlights
class TextCNN(nn.Module):
    def __init__(self, vocab_size, embedding_dim, num_classes, filter_sizes, num_filters,
dropout):
        super(TextCNN, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embedding_dim)

        # Multiple filter sizes to capture different n-gram patterns
        self.convs = nn.ModuleList([
            nn.Conv2d(1, num_filters, (filter_size, embedding_dim))
            for filter_size in filter_sizes
        ])

        # Dropout for regularization
        self.dropout = nn.Dropout(dropout)
        self.fc = nn.Linear(len(filter_sizes) * num_filters, num_classes)
# Bi-LSTM architecture highlights
class BiLSTMModel(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim, num_classes, dropout):
        super(BiLSTMModel, self).__init__()

        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        # Bidirectional LSTM to capture context from both directions
        self.lstm = nn.LSTM(
            embedding_dim,
            hidden_dim,
            bidirectional=True,
            batch_first=True
        )
        # Concatenating forward and backward hidden states
        self.fc = nn.Linear(hidden_dim * 2, num_classes)
        self.dropout = nn.Dropout(dropout)
```

4.3 Directions for Improvement

1. Data Augmentation: Techniques like back-translation or synonym replacement could help address the class imbalance issue, particularly for minority categories like "stock".
2. Pre-trained Word Embeddings: Incorporating pre-trained Chinese word embeddings (e.g., Chinese BERT or Word2Vec trained on a larger corpus) could improve the semantic representation of words.
3. Model Ensemble: Combining predictions from multiple models could leverage the strengths of different approaches and potentially improve overall accuracy.
4. Feature Engineering: Incorporating the news keywords in addition to titles could provide more semantic information for classification.
5. Advanced Architectures: Experimenting with transformer-based models like BERT or RoBERTa, which have shown state-of-the-art performance in text classification tasks.
6. Hierarchical Classification: Implementing a hierarchical classification approach might better handle the confusion between semantically related categories.
7. Title Length Considerations: Developing specialized models for extremely short titles, which might benefit from different feature extraction techniques.

In conclusion, while the Bi-LSTM model achieved promising results, there is still room for improvement through more sophisticated architectures and better handling of the class imbalance and semantic similarity challenges inherent in news classification tasks.