Reproducing FastText Sentiment Analysis

Using VADER Sentiment

Tuan Minh Luu

Macquarie University

October 20, 2024

Outline

1 Introduction

2 Replication of Original Work

3 Data Pre-processing

4 Model Training and Evaluation

5 Hyperparameter Tuning

6 Dataset Balancing

7 Comparison of Results

8 Model Utilities

9 Model Limitations and Future Improvements

10 Incorporating Techniques and Tools

11 Conclusion

Introduction

In this presentation, we describe the sentiment analysis project using FastText. The tasks involved include:

- Preprocessing and cleaning the BBC news dataset.
- Applying VADER sentiment analysis.
- Training a FastText model.
- Improving model performance with hyperparameter tuning.
- Balancing the dataset and re-evaluating the model.

Replication of Original Work

In this project, we started by reproducing the language identification task using fastText based on the tutorial by Edouard Grave. This followed the original work from the paper titled *"Bag of Tricks for Efficient Text Classification"* by Joulin, Grave, Bojanowski, and Mikolov (2017).

- Downloaded and built fastText from the official repository.
- Prepared datasets (bbc$_n$*ews*)*andtrainedaninitialmodel*.*Achievedanaccuracyof*95.3%*ontheva*

- This successful replication provided the foundation for extending the model to include Kazakh language support and other enhancements.

Data Preprocessing

The preprocessing steps applied to the dataset were as follows:

- Lowercasing and removing special characters.
- Removing stop words using the NLTK library.
- Lemmatizing text to its root form.

The cleaned data columns include:

- Title cleaned.
- Description cleaned.
- Combined sentiment (using VADER analysis).

Model Training and Evaluation

We utilized `FastText` for the training and evaluation process:

- **Training Parameters:**

```
model = fasttext.train_supervised(
        input ='bbc_news_train_combined.txt',
                 l.r=1.0, epoch=25,wordNgram=2,
                                minCount=1, verbose=2)
```

- **Evaluation Result:**
  - Precision: 73.68%
  - Racall: 73.68%
  - Confusion Matrix: Highlighted True Positive and False Negative.

Hyperparameter Tuning

A grid search approach was employed for hyperparameter tuning:

- Learning rates: 0.01, 0.05, 0.1
- Epochs: 25, 50
- WordNgrams: 1, 2
- Embedding Dimensions: 50, 100

The best model achieved a precision of 0.733 using:

- Learning rates: 0.01
- Epochs: 50
- WordNgrams: 2
- Embedding dimension: 100

The result of the best model is lower than the official model.

Dataset Balancing

Oversampling techniques were applied to balance the neutral class, avoid bias in model training. Class distribution after oversampling:

- Negative: 17801
- Positive: 14401
- Neutral: 14401

Model trained on the balanced dataset achieved a precision of 98.1%.

Comparison of Model Results

We compared the results of our trained model with the official fastText models using **precision** as the evaluation metric:

```
    test\_data['balanced\_predictions'] = test\_data['
        text\_only'].apply(lambda x: balanced\_model.
        predict(x)[0][0])
```

- **Our Model:** Precision = 98.1%
- **Official Model**: Precision = 73.68%
- **Official Model with hyperparameter:** Precision = 73.39%

Precision was chosen as it measures the proportion of correctly identified languages out of the total predicted. It is a crucial metric when accuracy alone doesn't capture performance in multi-class classification.

Model Utilities

The FastText-based sentiment analysis model offers several key utilities:

- **Efficient Sentiment Analysis:** Automates the classification of news articles, providing quick and scalable sentiment analysis for large datasets.
- **Real-Time Classification:** FastText allows for real-time sentiment detection, which is crucial for media monitoring platforms and sentiment-based recommendations.
- **Balanced Dataset Approach:** By employing oversampling, the model handles underrepresented classes effectively, ensuring reliable results across sentiment categories.
- **Generalizability Across Topics:** Demonstrates effective sentiment classification across diverse news topics, useful for media analysts and researchers.
- **Practical Applications for Media Organizations:** Helps automate content tagging, monitor sentiment trends, and analyze public opinion in real-time.

Model Limitations and Future Improvements

Despite achieving high accuracy, the model exhibits several limitations:

- **Overfitting Risk:** High accuracy on the test set could indicate overfitting, especially when the training and testing datasets are closely related.
- **Imbalance in Sentiment Classes:** Although oversampling was employed, the model may still face challenges in identifying minority classes like neutral sentiments.
- **Handling Ambiguous Sentiments:** The model struggles with articles that have mixed or subtle sentiments, leading to misclassification in cases where sentiments are not clear-cut.
- **NaN Errors During Training:** Several hyperparameter combinations resulted in NaN errors, limiting the effectiveness of hyperparameter tuning and indicating potential instability.
- **VADER Limitations:** Using VADER as the sentiment analysis base might be insufficient for complex language structures, leading to inaccuracies in cases involving nuanced language.

Incorporating Techniques and Tools

This project effectively utilized various techniques and tools to accomplish sentiment analysis on the BBC news dataset:

- **FastText for Text Classification:** Leveraged the FastText library for efficient model training and hyperparameter tuning to perform text classification tasks.
- **Data Cleaning and Preprocessing:** Employed regular expressions, NLTK's stopwords, and VADER sentiment analysis to clean and preprocess text data effectively.
- **Handling Class Imbalance:** Applied oversampling techniques to address the class imbalance challenge in the dataset, improving model performance.
- **Model Evaluation using Scikit-learn:** Evaluated model performance using metrics like precision, recall, F1-score, and confusion matrix to assess results.
- **Google Colab for Execution:** Leveraged Google Colab's cloud-based environment for running and testing large-scale experiments conveniently.

Conclusion

Key Takeaways:

- **Improved Sentiment Classification:** Achieved high precision through effective preprocessing and FastText training.
- **Class Imbalance Addressed:** Applied oversampling to enhance model accuracy, particularly for neutral sentiments.
- **Thorough Data Cleaning:** Used stopword removal, lemmatization, and filtering to refine sentiment predictions.

GitHub Repository: The repository contains:

- Python scripts for data cleaning and model training, training on balanced datasets and tuning hyperparameters.
- Instructions on reproducing results and applying improvements.