Sentence Classification using LSTMs and Baselines

HW1.B
Ashkan Ansarifard
1970082

Abstract

This report presents a sentence classification task using LSTM-based deep learning models, comparing their performance against traditional baselines. We evaluate Random, Majority, Stratified, and Word2Vec-based baselines on the HASPEEDE dataset and demonstrate the effectiveness of our LSTM models. Our results show that while baselines provide a simple benchmark, LSTMs significantly outperform them.

1 Introduction

Sentence classification is an important task in NLP, used in sentiment analysis, hate speech detection, and topic categorization. This paper explores LSTM-based models for classification and evaluates their performance against traditional baselines. We use the HASPEEDE dataset and measure the effectiveness of each approach.

2 Dataset and Preprocessing

The dataset consists of three sets: **train-taskA.jsonl** (training set), **test-news-taskA.jsonl** (news test set), and **test-tweets-taskA.jsonl** (tweets test set). We preprocess the text by tokenizing and creating a vocabulary from the training set, ensuring unknown words in the test set are mapped to a `<UNK>` token.

3 Baselines

We implement the following baselines:

- **Random Baseline**: Randomly assigns a class.
- **Majority Baseline**: Always predicts the most frequent class.
- **Stratified Baseline**: Predicts based on class distribution.
- **Word2Vec Baseline**: Uses Word2Vec embeddings to compute similarity with labeled training sentences.

4 Model Architecture

We implemented three versions of an LSTM model:

- **Simple LSTM**: A basic LSTM classifier with an embedding layer, a single LSTM layer, and a fully connected output layer.
- **Optimized LSTM**: Added batch normalization and dropout to improve generalization.
- **Parameterized LSTM**: Designed to test different hyperparameter settings to identify the best-performing model.

Each model uses a softmax or log-softmax output to classify sentences into predefined categories.

5 Design Choices

Several design choices were made to optimize model performance:

- **BiLSTM vs. LSTM**: Using BiLSTMs improved accuracy by capturing contextual information from both directions.
- **Batch Normalization**: Helped stabilize training and improved convergence speed.
- **Dropout Regularization**: Prevented overfitting, with an optimal dropout rate found at 0.3.
- **Word2Vec Baseline**: Implemented as an additional baseline to compare embeddings-based classification performance.

6 Instructions to Run the Code

6.1 Training a Single Model

To train a single LSTM model and save it:

```
python hw1b_train.py
```

This will store the model in `models/lstm_model.pth`.

6.2 Evaluating a Trained Model

To evaluate a trained model on test sets and compute baseline performances:

```
python hw1b_evaluate.py
```

This generates `results/evaluation_results.json`.

6.3 Running Multiple Experiments

To run multiple experiments with different LSTM configurations:

```
python pipeline.py
```

This saves trained models and results in timestamped directories inside `experiments/`.

7 Experiments and Results

We train and evaluate our models on the test sets. Table 1 presents the comparison of baselines and our best-performing LSTM model.

Model	Accuracy	Precision	Recall	F1-score
Random Baseline	49.6%	49.3%	49.2%	48.5%
Majority Baseline	63.8%	31.9%	50.0%	38.9%
Stratified Baseline	49.0%	43.6%	43.9%	43.7%
Word2Vec Baseline	57.8%	54.7%	54.8%	54.7%
LSTM Model	**66.4%**	**62.9%**	**61.7%**	**62.0%**

Table 1: Performance comparison on the News test set.

8 Conclusion

LSTM models outperform traditional baselines in sentence classification. While the Word2Vec baseline improves over traditional baselines, LSTMs provide the best results. Future work can explore transformer-based architectures or attention mechanisms to further improve classification performance.