Assignment 4

Natural Language Understanding (CSL7640)
Semester: AY 2024–25, Semester – II

Sentiment Analysis
REPORT

Under the guidance of

Prof. Asif Ekbal
Department of Computer Science and Engineering
Indian Institute of Technology Jodhpur

Submitted by:

Denzel Lenshanglen Lupheng (M24CSA036)
Jyotishman Das (M24CSA013)
Suvadip Chakraborty (M24CSA032)
Mehul Sah (M24CSA038)

Department of Computer Science and Engineering
Indian Institute of Technology Jodhpur

March 2025

Contents

0.1 Introduction

Sentiment analysis is a key task in natural language processing with applications in product reviews, social media, and opinion mining. In this report, we investigate two neural network approaches for sentiment classification:

- **Feed-Forward Neural Network (FFNN):** Applied to the IMDB dataset (binary classification) and an exploratory analysis on a Twitter dataset (multiclass classification).
- **Recurrent Neural Network (RNN):** Specifically, an LSTM-based model applied for both binary and multi-class sentiment tasks on the IMDB, Sentiment140, and Twitter datasets.

Each model is evaluated with appropriate metrics and the training dynamics are visualized using loss and accuracy curves (or cross-validation results).

0.2 Dataset and Preprocessing

We experiment on three datasets:

- **IMDB Dataset:** Contains movie reviews labeled as positive (1) or negative (0).
- **Sentiment140 Dataset:** Contains tweets labeled for sentiment (using a subset of 100,000 samples for quicker experimentation).
- **Twitter Dataset:** A new dataset with tweets labeled as Positive, Negative, or Neutral. Due to the absence of headers in the original file, the columns are manually renamed.

For all datasets, preprocessing steps include tokenization (using spaCy), vocabulary construction (with a minimum frequency threshold of 5), and padding/truncation based on the average sentence length.

0.2.1 IMDB Dataset Preprocessing (FFNN and RNN)

```
# Download and extract IMDB dataset
!wget -O aclImdb_v1.tar.gz https://ai.stanford.edu/~amaas/data/sentiment/
    aclImdb_v1.tar.gz
!tar -xzf aclImdb_v1.tar.gz

# Load 50% subset of the training data
def load_imdb_subset(imdb_dir='aclImdb', subset_ratio=0.5):
    data, labels = [], []
    for split in ['train']:
        for label, sentiment in [('pos',1), ('neg',0)]:
            folder = os.path.join(imdb_dir, split, label)
            files = glob.glob(os.path.join(folder, '*.txt'))
            random.shuffle(files)
            subset_files = files[:int(len(files)*subset_ratio)]
```

```
            for file in subset_files:
                with open(file, 'r', encoding='utf8', errors='ignore') as
                    f:
                    data.append(f.read())
                    labels.append(sentiment)
    return data, labels

train_texts, train_labels = load_imdb_subset()
```

Listing 1: IMDB Data Loading and Preprocessing

Tokenization and vocabulary construction:

```
nlp = spacy.load("en_core_web_sm", disable=["parser", "ner", "tagger"])
def tokenize(text):
    return [token.text.lower() for token in nlp(text)]
tokenized_texts = [tokenize(text) for text in train_texts]

word_counter = Counter()
for tokens in tokenized_texts:
    word_counter.update(tokens)
vocab = {word for word, count in word_counter.items() if count >= 5}
PAD_TOKEN = "<PAD>"
UNK_TOKEN = "<UNK>"
vocab.update({PAD_TOKEN, UNK_TOKEN})
vocab = sorted(list(vocab))
word2idx = {word: idx for idx, word in enumerate(vocab)}
vocab_size = len(vocab)
```

Listing 2: Tokenization and Vocabulary Building for IMDB

0.2.2 Sentiment140 and Twitter Dataset Preprocessing

Both datasets follow a similar preprocessing pipeline:

```
# Load Sentiment140 using a subset (100k samples)
def load_sentiment140(csv_path, max_samples=100000):
    texts, labels = [], []
    with open(csv_path, "r", encoding="latin-1") as csvfile:
        reader = csv.reader(csvfile)
        count = 0
        for row in reader:
            try:
                target = int(row[0])
                text = row[5]
                texts.append(text.strip())
                labels.append(1 if target == 4 else 0)
            except Exception:
                continue
            count += 1
            if count >= max_samples:
                break
    return texts, labels
```

Listing 3: Sentiment140 Preprocessing (subset of 100k samples)

```
import pandas as pd
twitter_file = 'twitter_training.csv'
df_twitter = pd.read_csv(twitter_file)
df_twitter.columns = ["id", "category", "sentiment", "tweet_text"]

sentiment_map = {"Negative": 0, "Neutral": 1, "Positive": 2, "Irrelevant": 3}
df_twitter['label'] = df_twitter['sentiment'].map(sentiment_map)
df_sample = df_twitter.sample(frac=0.3, random_state=42).reset_index(drop=True)
nlp = spacy.load("en_core_web_sm", disable=["parser", "ner", "tagger"])
def tokenize(text):
    return [token.text.lower() for token in nlp(text)]
tokenized_texts = [tokenize(text) for text in df_sample['tweet_text'].tolist()]
```

Listing 4: Twitter Dataset Preprocessing

0.3 Methodology

The experiments are carried out using two neural network architectures:

0.3.1 Feed-Forward Neural Network (FFNN)

The FFNN model uses an embedding layer to convert token indices into 300-dimensional dense vectors. These embeddings are averaged (simple average pooling) to create a fixed-size representation of each review or tweet. The resultant representation passes through two fully connected layers before producing a binary (for IMDB) or multiclass (for Twitter) output.

```
class FeedForwardNN(nn.Module):
    def __init__(self, vocab_size, embed_dim=300, hidden1=256, hidden2=128):
        super(FeedForwardNN, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim, padding_idx=word2idx[PAD_TOKEN])
        self.fc1 = nn.Linear(embed_dim, hidden1)
        self.fc2 = nn.Linear(hidden1, hidden2)
        self.fc3 = nn.Linear(hidden2, 1)  % Binary classification output
        self.activation = nn.ReLU()
    def forward(self, x):
        emb = self.embedding(x)
        pooled = emb.mean(dim=1)
        x = self.activation(self.fc1(pooled))
        x = self.activation(self.fc2(x))
        x = self.fc3(x)
        return x
```

Listing 5: Feed-Forward Neural Network Model (IMDB)

The training loop logs train loss and accuracy over 10 epochs. Sample inference is performed on random examples from the development split.

0.3.2 Recurrent Neural Network (RNN) with LSTM

The RNN model is defined using a single LSTM layer. Dense embeddings of tokens (300 dimensions) are fed to the LSTM (with a hidden size of 256). The final hidden state is then passed to a linear output layer to produce logits. This model is applied to all three datasets (IMDB, Sentiment140, and Twitter) with appropriate adjustments for the number of classes.

```
class SentimentLSTM(nn.Module):
    def __init__(self, vocab_size, embed_dim=300, hidden_dim=256,
        output_dim=1):
        super(SentimentLSTM, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim, padding_idx=
            word2idx[PAD_TOKEN])
        self.lstm = nn.LSTM(embed_dim, hidden_dim, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_dim)
    def forward(self, x):
        embedded = self.embedding(x)
        _, (h_n, _) = self.lstm(embedded)
        last_hidden = h_n[-1]
        out = self.fc(last_hidden)
        return out
```

Listing 6: RNN Model (LSTM) for Sentiment Analysis

For multiclass tasks (e.g., Twitter with 3 classes) the output layer's dimension is set to 3 and Cross-Entropy Loss is used.

0.4 Results

0.4.1 Feed-Forward Neural Network Experiment

IMDB Dataset (Binary Classification)

Figure **??** shows the combined plot of training loss and dev accuracy (logged per epoch) for the IMDB dataset using the FFNN model.

Twitter Dataset (Multiclass Classification)

An exploratory analysis using the FFNN model on a subsample of the Twitter dataset is conducted. Descriptive statistics, sentiment distributions, and histograms of tweet lengths are provided (see Figures **??**–**??**). Additionally, sample tweets are manually inspected.

0.4.2 Recurrent Neural Network Experiment

IMDB Dataset (Binary Classification)

Figure 1 displays the training loss and dev accuracy over 10 epochs for the RNN model (LSTM) applied on the IMDB dataset.

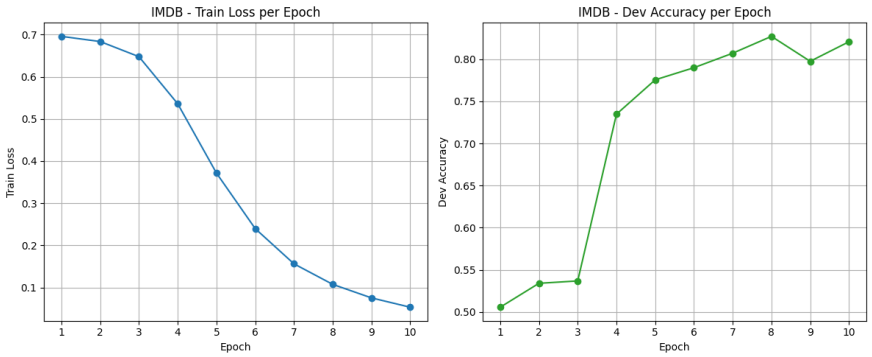

Figure 1: IMDB Dataset (RNN): Training Loss and Dev Accuracy vs. Epoch.

Sentiment140 Dataset (Binary Classification)

Figure 2 presents the combined plot for the Sentiment140 dataset (100k samples) using the RNN model. Here, the dev accuracy remains at 1.0 across epochs, indicating potential overfitting.

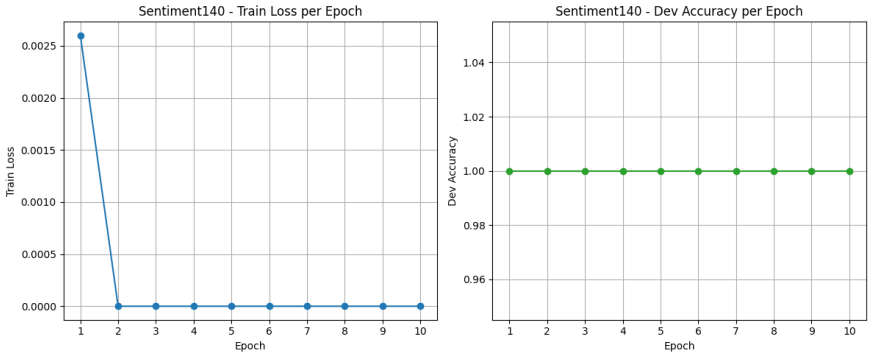

Figure 2: Sentiment140 Dataset (RNN): Training Loss and Dev Accuracy vs. Epoch.

Twitter Dataset (Multi-class Classification)

For the Twitter dataset, 5-fold cross-validation is employed. Figure 3 shows the accuracy for each fold, with an average accuracy of approximately 89%.

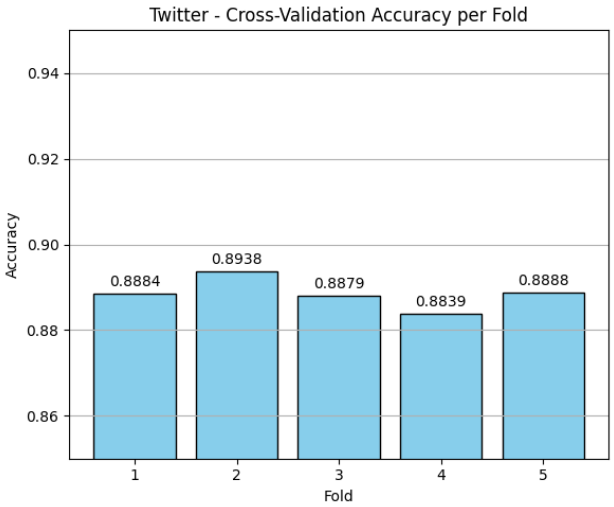

Figure 3: Twitter Dataset (RNN): Cross-Validation Accuracy per Fold.

0.5 Discussion

The FFNN experiments on the IMDB dataset demonstrate steady improvements in training loss and dev accuracy, indicating effective learning despite the simplicity of average pooling. In contrast, the RNN model shows similar trends on IMDB but with different convergence characteristics. On Sentiment140, both models report perfect dev accuracy on a small subset, suggesting overfitting or issues in sample representativeness. The Twitter dataset experiments, conducted via cross-validation, yield an average accuracy around 89% for the RNN, highlighting its potential for handling multi-class sentiment tasks.

0.6 Conclusion

This report has presented experiments using both Feed-Forward and Recurrent Neural Network models for sentiment analysis on multiple datasets. The FFNN approach provides a baseline, while the RNN (LSTM) model exhibits comparable performance with different convergence dynamics. The disparities across datasets underscore the importance of dataset size and representativeness. Future work will include a deeper investigation of preprocessing techniques and the incorporation of additional architectures for comparative analysis.