Assignment 4: Text Data

Git link:
https://github.com/JahnaviPokala/jpokala_64061/tree/main/Assignment%204%3A%20Text%20Data

Overview of the Assignment:
The objective of this task is to explore the application of Recurrent Neural Networks (RNNs) in text and sequence data analysis, using the IMDB sentiment classification dataset as an example. The task is intended to understand the impact of data limitations on model performance and evaluate approaches to enhance predictive accuracy through embedding techniques. Specifically, we train and adapt an existing RNN model to work on truncated reviews, a reduced training set, and a reduced vocabulary, and evaluate both pretrained and learned word embeddings. Through this exercise, we aim to find effective modeling techniques when presented with limited textual datasets and determine the suitability of different embedding techniques to improve model performance.

Defining the Task:

For natural language processing tasks such as sentiment analysis, data quantity and quality significantly affect the performance of deep models such as Recurrent Neural Networks (RNNs). This assignment will investigate the challenge of training RNNs from limited text data and compare multiple approaches to improving the accuracy of the model under such constraints. Specifically, we approach the challenge of building a robust sentiment classifier from the IMDB dataset with the following constraints in place: only training on a small sample of 100 reviews and input sequences truncated to 150 words. In addition, we contrast model performance trained using standard embedding layers with that trained using pre-trained word embeddings in order to determine which method results in better generalization on out-of-sample data.

Key Goals:

- Use Recurrent Neural Networks (RNNs) to perform sentiment analysis on text data from the IMDB movie reviews dataset.
- Test the impact of data limitations by training the model using a reduced dataset of only 100 samples and reviews truncated at 150 words.
- Compare embedding methods, including a standard trainable embedding layer and pretrained word embeddings, to determine which approach leads to better performance.
- Analyze validation accuracy patterns with different training set sizes to find at which size learned embeddings are most effective.

- Draw conclusions regarding model design choices that perform best in scenarios where labeled data is lacking for text classification tasks.

Dataset Summary:

The dataset for this task is the IMDB Movie Reviews dataset with its 50,000 positive or a negative review overall. The dataset itself is balanced, meaning it has the same number of positive and negative reviews. For the purpose of this task: We're only considering the top 10,000 most frequent words from the vocabulary. Reviews are cut off after 150 words, for the sake of consistency and to try to keep the sequence length reasonable. The training dataset will only consist of up to 100 samples, while the validation dataset is given as 10,000, as stated in the assignment requirement.

Data Preparation:

1. Tokenization and Vectorization:

IMDB dataset, already in tokenized form as a list of integers, was truncated to the top 10,000 words using num_words=10000.

Sequence Truncation and Padding:

Reviews were truncated to the first 150 tokens, and the sequences were padded with pad_sequences() to equal length.

2. Dataset Splitting:

• Training set: First 100 samples

• Validation set: First 10,000 samples of the test set

3. Embedding Options:

Two embedding options were employed:

• A trainable Embedding layer initialized randomly.

• A pre-trained embedding (e.g., GloVe), loaded and utilized through a non-trainable Embedding layer (as illustrated in the notebook).

Model Design:

Two prevalent model variations were employed:

1. Trainable Embedding Model

Embedding Layer: Trainable, randomly initialized

Bidirectional LSTM: Bidirectional dependencies are modeled

Dense Output Layer: A single neuron with sigmoid activation for binary output

2. Pretrained Word Embedding Model

Embedding Layer: Pretrained weights (GloVe vectors) initialized, non-trainable

Bidirectional LSTM: Pretrained weights (GloVe vectors) initialized, non-trainable

Dense Output Layer: Pretrained weights (GloVe vectors) initialized, non-trainable

Performance Review:

Here, we evaluate the performance of the RNN models trained on a small dataset using two embedding techniques: a trainable embedding layer and a pre-trained word embedding (e.g., GloVe). The models are evaluated on a validation set of 10,000 IMDB reviews.

1. Model with Trainable Embedding Layer

Training Samples: 100

Validation Accuracy: Medium (typically ~69%)

The trainable embedding model learned embeddings from scratch, which was less effective given the small dataset size. The small data limited the model's ability to generalize patterns and relations between words.

2. Model with Pretrained Word Embeddings

Training Samples: 100

Validation Accuracy: Higher than the trainable embedding model (usually ~84% or higher)

Using pretrained embeddings allowed the model to take advantage of semantic knowledge learned from a significantly larger corpus (e.g., GloVe). This improved performance, especially with very limited amounts of training data.

3. Effect of Training Sample Size

A more extended experiment was conducted by changing the size of the training samples (e.g., 100, 500, 1000, 2000, etc.) to determine at which point the trainable embedding starts to perform as well as or better than the pretrained embedding.

For <1000 samples, pretrained embeddings performed better consistently.

As training size increased above ~2000–3000 samples, the trainable embeddings caught up and did slightly better in some runs as they were able to fine-tune and specialize for the IMDB domain.

4. Performance Visualization

Models Built	Training sample size	Validation size	Testing sample size	Model Type	Test accuracy (%)	Test loss
Model 1	100	10000	5000	embedded layer	69.3	50.7
Model 2	7000	10000	5000	embedded layer	86.8	31
Model 3	12000	10000	5000	embedded layer	88.4	27.3
Model 4	10000	10000	5000	one-hot	84.6	36.3
Model 5	15000	10000	5000	LSTM using embedded layer	50.9	69.3
Model 6	25000	10000	5000	LSTM using embedded layer	50.4	69.3
Model 7	25000	10000	5000	masking enable	96	53.7
Model 8	100	10000	5000	Pretrained using	50.8	69.6
Model 9	9000	10000	5000	Pretrained with 4 LSTM hidden layers	60.4	43.7
Model 10	20000	10000	5000	Pretrained with 2 LSTM hidden layers	65.7	47.6

Conclusion:

This task demonstrated the application of Recurrent Neural Networks (RNNs) for sentiment analysis on the IMDB dataset with focus on handling limited data scenarios. By comparing two embedding strategies, the traditional trainable embedding layer and pretrained word embedding—we were able to draw useful conclusions regarding how each performs under constraints.

The experiments show explicitly that pretrained embeddings perform much better than trainable embeddings when the training set is small because they are already exposed to large text sets and semantic depth prior to it. However, with the growth in the size of the training set, trainable embeddings begin improving and become equivalent in performance.

Also, the use of Bidirectional LSTM layers proved good in preserving contextual information in sequences of text as well, which resulted in higher classification accuracy.

Overall, this experiment highlights the importance of integrating selection and data availability into text classification model design. Pretrained embeddings are especially valuable in low-data scenarios, which makes them a preference choice in real-world applications where annotated data is limited.