PROJECT REPORT

INFORMATION RETRIEVAL PROJECT:

Sentiment Analysis via Neural Networks

Instructor: Sir Rafi

IR PROJECT REPORT

FAST-NUCES BSCS SPRING SEMESTER SECTION 6A

OVERVIEW

1. Group Members

21K-4676 Maryam Shahid

21K-3170 Shaheer Badar

2. Overview

Sentiment analysis is a vital task in natural language processing (NLP) that involves determining the sentiment expressed in a piece of text, and categorizing it as positive or negative. In this project, we developed and compared three different models—Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Latent Dirichlet Allocation (LDA)—for sentiment analysis using the Large Movie Review Dataset v1.0 and implemented accordingly the Research papers 'Performance Analysis of Different Neural Network for Sentiment Analysis on IMDb Movie Reviews' and 'Learning Word Vectors for Sentiment Analysis' as provided.

3. Model and Libraries

In this project, we implemented three models using Python: CNN, LSTM, and LDA. Below is a brief description of each model and the libraries used.

Convolutional Neural Network (CNN)

CNNs have proven to be effective in text classification due to their ability to capture local features through convolutional layers.

The architecture includes an embedding layer followed by a convolutional layer with 30 filters and a kernel size of 3, which helps in extracting local features from the text. A max pooling layer is used to reduce feature maps, and then a flatten layer to convert features of n-dimension into a 1D vector. Finally, two dense layers are used, with the last layer utilizing a sigmoid activation function to output a binary sentiment prediction.

The model is trained for 8 iterations with a batch size of 128, having 196 iteration in each epoch. After training, the function predicts the sentiment labels for the test set by converting the predicted probabilities into binary labels based on a threshold of 0.5.

Libraries Used:

- `numpy`: Used for storing and processing large dimensional arrays.
- `sklearn.metrics`: Used its functions to compute accuracy, precision, and recall.
- `tensorflow.keras.models`: Used Sequential() - a built in class that provided stack of layers for building and load_model() - for loading models.
- `tensorflow.keras.layers`: Provides various neural network layers, including Dense, Embedding, Conv1D, MaxPooling1D, and Flatten.
- `tensorflow.keras.preprocessing.sequence`: Contains functions for sequence preprocessing, including padding sequences –used to make the review data of the same dimension while training the model.

Long Short-Term Memory (LSTM)

LSTM well-known for training models for sequential data like text due to their ability to capture long-term dependencies.

LSTM includes an embedding layer to convert words into dense vectors of fixed size, an LSTM layer with 100 units to capture long-term dependencies in the training reviews. Finally, two dense layers are used, with the last layer employing a sigmoid activation function to output a binary sentiment classification, and compiled using Adam optimizer.

The model is trained for 5 epochs with a batch size of 128. After training, the function predicts the sentiment labels for the test set by converting the predicted probabilities into binary labels based on a threshold of 0.5.

Libraries Used:

- `numpy`, `sklearn.metrics`.
- `tensorflow.keras.models, tensorflow.keras.preprocessing.sequence.`
- `tensorflow.keras.layers`: Provides various neural network layers, including Dense, Embedding, LSTM.

Latent Dirichlet Allocation (LDA)

LDA is a generative probabilistic model used for topic modeling and discovering abstract topics within a collection of documents. LDA can be adapted for sentiment analysis.

The LDA model is trained on the dataset to learn the topic distributions, which are then saved using joblib. The document-topic distributions are extracted using the trained LDA model. To incorporate sentiment information, a custom function incorporate_sentiment is defined, which adjusts the topic-word distributions based on the sentiment labels. This function calculates sentiment distributions for each topic and adjusts the topic-word distributions accordingly, enhancing the representation of words based on sentiment. The adjusted topic-word distributions are normalized to ensure they sum to one for each topic.

Libraries Used:

- `os`, `numpy`
- `sklearn.feature_extraction.text`: Provides functions for text feature extraction, including CountVectorizer.
- `sklearn.decomposition`: Provides decomposition algorithms, including Latent Dirichlet Allocation.
- `joblib`: Used for serializing Python objects.

4. Results:

Each model is evaluated using the metrics of accuracy, precision and recall. Below are the results:

CNN Results

- **Accuracy**: 0.8607
- **Precision**: 0.8900
- **Recall**: 0.8232

LSTM Results

- **Accuracy**: 0.8689
- **Precision**: 0.8937
- **Recall**: 0.8374

LDA Results

- **Accuracy**: 0.7700
- **Precision**: 0.7600
- **Recall**: 0.7800

5. Conclusion:

Each model has its strengths and weaknesses, with CNN and LSTM being more suitable for sequence data and LDA offering a different approach through topic modeling. The results indicate that neural network-based models (CNN and LSTM) generally perform better than LDA for sentiment classification tasks.

6. Reference Research Papers:

wvSent_acl2011.pdf

ICECTE-PID-209.pdf