

Daniel Choy
Spring 2023
CSE 144

Assignment 3 RNN for Natural Language Processing **(Report)**

Preprocessing Techniques:

BERT Pre-Trained Transformer Model

To preprocess the data, I tokenized and converted the text to numerical representations using word embeddings. Once this was done, I split the dataset into train and eval datasets and shuffled them.

Logistic Regression Model

To preprocess the data, I tokenized the text and converted them into pandas data frames. Then I converted the tokenized text to bag of words representation. Once this was done, I split the dataset into train and eval datasets and shuffled them.

Techniques

Tokenization:

- Tokenization is the process of splitting text into smaller units called tokens. In the context of natural language processing (NLP), tokens typically represent individual words.

Word Embedding:

- Word Embedding represents words or phrases as dense, low-dimensional vectors in a continuous space in order to capture relationships between words based on their contextual usage. Similar words are located closer to each other.

Bag of Words:

- Bag of Words representation converts texts into numerical vectors. In this approach, a collection of text documents is represented as a collection of unique words or the “vocabulary.” Each document is then represented as a vector, where each element of the vector corresponds to the count or presence of a word in the document.

Splitting Dataset into Train and Eval Datasets:

- By splitting the dataset into train and eval datasets, I have a dataset to train the model and tune hyperparameters with and a dataset to evaluate the trained model's performance on unseen data. Splitting the dataset into train and eval subsets enables a more robust and reliable evaluation of the model's performance, allowing for better model development.

Shuffling Datasets:

-Shuffling the datasets help to eliminate biases, improve generalization, prevent overfitting, and ensure consistency in the learning process.

Model Architecture:

BERT Pre-Trained Transformer Model

BERT is a pre-trained transformer model in which its architecture consists of two main components.

The first is that it is based on a multi-layer transformer encoder which is composed of a stack of identical layers, each containing a self-attention mechanism and a position-wise feed-forward neural network. The self-attention mechanism allows the model to attend to different words in the input sequence to capture contextual dependencies. The position-wise feed-forward network applies a non-linear transformation to each position separately.

The second is that it is pre-trained on large-scale unlabeled text data using two self-supervised learning tasks called masked language modeling and next sentence prediction. Masked language modeling involves randomly masking some words in the input sequence and training the model to predict the masked words. Next sentence prediction involves predicting whether two sentences appear consecutively in the original text or not.

Logistic Regression Model

The architecture of the Logistic Regression model in scikit-learn involves three layers. The first layer is the input layer which represents the input features. The second layer is the linear combination layer which computes the linear combination of the input features and their corresponding weights. It multiplies each input feature by its weight and sums up the products. The third layer is the activation layer which applies the sigmoid activation function to the linear combination computed in the previous layer. The logistic function transforms the linear combination into a value between 0 and 1, representing the predicted probability of the positive class.

Hyperparameter Tuning:

BERT Pre-Trained Transformer Model

```
batch_size = 10
num_epochs = 7
learning_rate = 0.0001
```


Dataset Size for both Models (BERT and Logistic)

```
# Split into train and eval datasets and shuffle them
train_dataset = tokenized_datasets["train"].shuffle(seed=42).select(range(3000))
eval_dataset = tokenized_datasets["test"].shuffle(seed=42).select(range(3000))
```

The size of the dataset for both the training and eval datasets was 3000. I noticed that as I increased the size, the accuracy got better for the Logistic model but not the BERT model and the computation time would increase for both models.

Model Evaluation:

BERT Pre-Trained Transformer Model



[2100/2100 46:43, Epoch 7/7]

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	1.308400	1.104245	0.516333	0.501039	0.516333	0.494909
2	1.006100	1.127172	0.512000	0.588346	0.512000	0.495156
3	0.778900	1.092295	0.560000	0.590372	0.560000	0.561983
4	0.515200	1.300106	0.560333	0.566667	0.560333	0.554981
5	0.316700	1.902629	0.544000	0.587623	0.544000	0.548753
6	0.165600	2.437926	0.565000	0.581950	0.565000	0.570156
7	0.060200	2.742888	0.560667	0.587502	0.560667	0.566518

TrainOutput(global_step=2100, training_loss=0.5930179532368978, metrics={'train_runtime': 2808.1716, 'train_samples_per_second': 7.478, 'train_steps_per_second': 0.748, 'total_flos': 5525480991744000.0, 'train_loss': 0.5930179532368978, 'epoch': 7.0})

Logistic Regression Model

```
Accuracy: 0.473
Precision: 0.46952058355870363
Recall: 0.473
F1-Score: 0.4710250316122826
```

Advantages of Deep Learning Models for Sentiment Analysis:

Deep learning models have many learning advantages. These models can learn complex patterns, can process text data in its original form, can learn word order and contextual information, and can learn semantic and syntactic information from word embeddings.

Another advantage of deep learning models is their ability to efficiently handle large-scale datasets. They can be trained on massive amounts of data, which is beneficial for sentiment analysis tasks that require a diverse/extensive training corpus.

Deep learning models not only perform much better than other types of models but they are also flexible/adaptable as they can be adapted and customized for specific sentiment analysis tasks and domains. They can be trained on domain-specific data to

capture domain-specific sentiment patterns and improve performance in specific contexts. Pre-trained models can be fine-tuned for a specific dataset and problem.

Limitations of Deep Learning Models for Sentiment Analysis:

Deep learning models typically require a large amount of labeled training data to perform well. Acquiring a sizable sentiment analysis dataset can be time-consuming and expensive.

Additionally, another limitation is that deep learning models are prone to overfitting, especially when training data is limited.

Also, pre-trained deep learning models may not effectively generalize to specific sentiment analysis tasks so fine-tuning or retraining on domain-specific data is often necessary to achieve optimal performance.

Lastly, while deep learning models can capture semantic relationships, they may still struggle with understanding sarcasm, irony, or cultural context which can significantly impact sentiment analysis.

Recommendations for Future Research:

- Explore techniques to improve sentiment analysis performance in specialized domains such as finance, healthcare, or legal domains.

- Extend sentiment analysis beyond binary classification (positive/negative) to more fine-grained levels.

- Investigate approaches to incorporate multiple modalities, such as text, images, audio, and video, for sentiment analysis.

- Develop models and techniques that can effectively detect and interpret sarcasm and irony in text.