PROJECT REPORT

Evaluation of various deep learning models for sentiment analysis

We are given the reviews dataset. There are 194439 amazon reviews for cell phones and accessories taken from https://jmcauley.ucsd.edu/data/amazon/ Use the “reviewText”, “summary” and “overall” fields from this file. The goal is to predict the rating given the review by modeling it as a multi-class classification problem.

Took the first 70% dataset for train, next 10% for validation/development, and remaining 20% for test.

Summary of ML Models after TFIDF:

ML Model	Train		Validate		Test	
	Accuracy	F1-Score(weighted)	Accuracy	F1-Score(weighted)	Accuracy	F1-Score(weighted)
Decision Trees	100	100	44	47	44	47
Linear SVM	57	54	63	62	64	64
Logistic Regression	57	56	62	63	63	64
RandomForest	100	100	58	58	58	59
XGBoost	66	65	61	61	62	63
SVM rbf	58	54	62	62	65	64

Word2vec Embeddings

The basic idea behind word2vec is to train a neural network to predict the context of each word in a large corpus of text. The network is trained to predict the probability of observing a particular context word given a target word. During this process, the network learns to represent each word as a high-dimensional vector in the vector space, such that words that are semantically similar are located close to each other in the vector space.

The two main variations of word2vec are the Continuous Bag-of-Words (CBOW) and the Skip-gram models. In CBOW, the network predicts the target word based on the surrounding context words, while in Skip-gram, the network predicts the context words based on the target word.

Word2vec embeddings have been shown to be effective in a variety of natural language processing (NLP) tasks, such as text classification, sentiment analysis, named entity recognition, and machine translation. The embeddings can also be visualized to reveal interesting semantic relationships between words.

WORD2VEC EMBEDDING	**Train**		**Validate**		**Test**	
vector, window, min_count	**Accuracy**	**F1-Score(weighted)**	**Accuracy**	**F1-Score(weighted)**	**Accuracy**	**F1-Score(weighted)**
100,3,2	61	56	63	58	66	60
100,3,5	61	56	63	58	65	60
100,7,2	62	56	64	58	66	60
100,7,5	62	57	64	59	66	61
200,3,2	62	57	64	59	66	61
200,3,5	62	57	64	59	66	61
200,7,2	63	58	64	59	66	61
200,7,5	63	58	64	59	66	61
300,3,2	63	58	64	59	67	62
300,3,5	63	58	64	59	66	61
300,7,2	63	58	64	60	67	62
300,7,5	63	58	64	60	67	62

GLOVE EMBEDDING:

GloVe aims to capture the meaning of words by looking at the co-occurrence statistics of words in a large corpus of text. The intuition behind this approach is that words that often appear together are likely to be related in meaning.

GloVe generates word embeddings by constructing a co-occurrence matrix based on the frequency of words appearing together in a given corpus. This matrix is then factorized using matrix decomposition techniques to obtain low-dimensional word vectors.

The resulting word vectors can be used in a variety of natural language processing tasks, such as text classification, machine translation, and sentiment analysis. They have been shown to outperform other popular word embedding methods such as Word2Vec and LSA (Latent Semantic Analysis) in some tasks.

GloVe embeddings are widely used and pre-trained versions of these embeddings are available for download and use in various NLP applications.

GLOVE EMBEDDING	Train		Validate		Test	
Dimensions	Accuracy	F1-Score(weighted)	Accuracy	F1-Score(weighted)	Accuracy	F1-Score(weighted)
50d	56	45	58	47	62	51
100d	57	49	59	51	62	53
200d	59	51	60	53	63	55
300d	59	52	61	54	64	56

RNN and LSTM:

A recurrent neural network (RNN) is a type of artificial neural network that is designed to process sequential data, such as time series, speech, and text. Unlike feedforward neural networks, which process input data only in one direction, from input to output, RNNs are designed to take into account previous inputs in the sequence, making them well-suited for tasks that require the analysis of patterns over time.

RNNs are based on the idea of sharing weights across time steps, which means that the same set of weights is used to process each input in the sequence. This allows the network to learn long-term dependencies between the inputs in the sequence, which can be crucial for many applications, such as speech recognition, natural language processing, and time series prediction.

One of the most commonly used variants of RNN is the Long Short-Term Memory (LSTM) network, which is designed to address the problem of vanishing gradients that can occur in traditional RNNs. LSTMs use a memory cell that can store information over long periods of time, as well as gates that control the flow of information into and out of the cell, making them better suited for tasks that require the processing of long-term dependencies.

RNN/LSTM	Train		Validate		Test	
	Accuracy	F1-Score(weighted)	Accuracy	F1-Score(weighted)	Accuracy	F1-Score(weighted)
RNN_1	62	60	63	61	62	62
RNN_2	57	62	61	54	62	61
RNN_3	61	61	64	59	63	62
RNN_4	63	60	64	59	64	62
LSTM_1	63	62	64	59	62	62
LSTM_2	62	61	63	58	63	61
LSTM_3	62	60	63	58	63	62
LSTM_4	62	62	62	62	63	62

BERT:

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model that can be fine-tuned for various natural language processing (NLP) tasks such as text classification, named entity recognition, and question answering.

For classification tasks, BERT can be used by adding a classification layer on top of the pre-trained BERT model. The input text is tokenized, and each token is converted to a vector representation using the pre-trained BERT model. These vector representations are then fed into the classification layer, which produces the final output.

When fine-tuning BERT for classification tasks, the pre-trained model's parameters are updated using labeled data specific to the task at hand. This fine-tuning allows the model to learn task-specific features and achieve higher performance.

To use BERT for text classification, you can follow these steps:

1. Load the pre-trained BERT model
2. Tokenize the input text using the BERT tokenizer
3. Add special tokens [CLS] at the beginning of the input text and [SEP] at the end
4. Convert the tokenized input to input IDs and attention masks
5. Fine-tune the BERT model for the specific classification task
6. Evaluate the performance of the model on the validation set
7. Use the model to make predictions on the test set.

Overall, BERT has achieved state-of-the-art performance on many classification tasks, and it is widely used in industry and academia for NLP tasks.

	Train		**Validate**		**Test**	
BERT	**Accuracy**	**F1-Score(weighted)**	**Accuracy**	**F1-Score(weighted)**	**Accuracy**	**F1-Score(weighted)**
BERT	62	60	63	61	62	62