CSC 693 - HW 3 - Deep Learning

Sahil Dhawan

I. Sentiment Analysis

Sentiment analysis is the process of analyzing text to determine the emotional tone. Use all the 2,000 comments in Dataset1 as the corpus, and choose related python libraries to finish the following tasks.

A. Part 1)

In Assignment2, you developed a neural network classifier C1 with the pre-trained word embedding model, now develop a new neural network classifier C2 without loading pre-trained word embeddings. Compare C2 and C1, which one has better performance? Try to discuss it. (Requirements: you developed three classifiers in Assignment2, pick the best one as C1. Make sure C1 and C2 are trained and evaluated with the same training, validation and testing sets).

- C1 Settings: word2vec CBOW with `vector_size=100, window=5, min_count=1`
- C2 Settings: No pretrained word embeddings, words were tokenzied with Tokenizer from tensorflow.keras.preprocessing.text and pad_sequence from tensorflow.keras.preprocessing.sequence

C1 vs C2	C1	C2
Accuracy	0.7800	0.5050
Precision	0.7849	0.4167
Recall	0.7526	0.0515
F1	0.7684	0.0917

C1 has better performance because of using pretrained word embeddings. Tokenizing words on their own to use as input to the neural network is not effective.

B. Part 2)

Train three classifiers with RNN, LSTM and GRU respectively under the same settings, and output the classification reports (tip: you can use sklearn.metrics.classification_freport). Which classifier has better performance? Compare their time cost, which one is faster? (Requirement: Use **Pytorch** for this problem. Clearly specify the dataset division, hyperparameters and other required settings in your report)

- Hyperparameters:
  - Learning Rate = 0.001
  - Batch Size = 64
  - 50 Epochs
  - 15 Hidden Layers
  - Adam Optimizer
  - BCELoss with Sigmoid Layer

Test Data Evaluation	RNN	LSTM	GRU
Accuracy	77.50%	79,00%	76.50%
Precision	75.49%	77.78%	77.17%
Recall	79.38%	79.38%	73.20%
F1	77.39%	78.57%	75.13%
Runtime (sec)	7.5873	3.2012	15.6563

Word embeddings were passed in as input data with the same settings as for C1 in Part 1/Assignment 2. The data was split into 80% training, 10% validation, and 10% testing. All three classifiers have nearly identical performance with LSTM having the highest accuracy, precision, and F1 score and RNN and LSTM having the same highest recall value. LSTM has the fastest compute time, followed by RNN and GRU. Because the data set of 2000 comments is relatively small for a deep learning architecture, the similarity in evaluation scores could be due to underfitting.

The architecture of RNN is similar to short-term memory, so gradients which are too small or too large can cause vanishing gradient or exploding gradient problems. Exploding gradients especially can become a problem because it can cause the classification to become biased.

LSTM's architecture resemebles a combination of short and long-term memory. GRU is a simplified variation of LSTM which has update and reset gates instead of an output gate and combines the cell and hidden state per each unit. Ideally, LSTM should have the best evaluation scores in addition to compute time because of its inclusion of cell states and hidden states which are passed into a sigmoid layer to determine if the values are important to keep or not.

C. Part 3)

Build a bidirectional 3-layer stacked LSTM model, compared to your LSTM model in the last question, which one has better classification results? Why?

Test Data Evaluation	LSTM	BD 3LS LSTM
Accuracy	79,00%	79.00%
Precision	77.78%	77.23%
Recall	79.38%	80.41%
F1	78.57%	78.79%
Runtime (sec)	3.2012	5.8805

Both models have similar results, but the bidirectional 3-layer stacked LSTM has a better recall. It is expected that bidirectional 3-layer stacked LSTM should perform better due to performing in two directions: ”past” and ”future”, creating a more comprehensive long and short term memory context of the data set.

post-answer update: *I noticed my code for rnn, lstm, and gru did not include hidden or cell state in the function definition when the type_name variable was set to 'lstm', and I could not get hidden and cell state to work correctly in the bidirectional 3-layer stacked LSTM, so the code did not use those variables.*

II. Text Translation

Machine translation is the process of automatically translating content from one language to another without any human input. Given the English-Spanish dataset2, build a machine translator using TensorFlow. (The format of the dataset is English + TAB + Spanish + TAB + CC-BY License + Attribution. You can basically ignore the license and attribution, only focus on the columns of English and Spanish.)

A. Part 1)

Prepossess the dataset and split the dataset into training, validation and testing subsets by the ratio 70/15/15. The english and spanish parsed lists each were length 118,629.

The data was parsed and passed into a .pkl file after the first parse to load in easier on each test run. Each entry of an english or spanish word was respectively matched with an 'english' or 'spanish' label as it was appended to the appropriate list.

B. Part 2)

Train the translator with Transformer method, report the classification results. What does the hyperparameter “num_heads” mean? Why do we need this mechanism?

num_heads is the number of times the self-attention mechanism is applied in parallel with different sets of learned parameters. Multi-head attention analyzes relationships between different input elements ”simultaneously”.

C. Part 3)

What is the translation result for the sentence “Deep Learning is widely used in Natural Language Processing, as Dr. Sun said in CSC 495/693.” with your translator? Plot the heatmap of the encoder-decoder attention scores.