CS 6320: Natural Language Processing

Project 1: Sentiment Classification with Deep Learning (100 points)

Due date: 10/25/2019 11:59 pm

1 Introduction

For Project 1, you will implement a deep learning model for performing sentiment classification of movie reviews.

2 Problem Definition and Dataset

Sentiment analysis or opinion mining refers to the task of classifying texts based on their affect i.e. whether the expressed opinion is positive, negative or neutral. Some advanced opinion mining tasks classify texts on the basis of mood (e.g. happy, angry, sad, etc.), expression (e.g. sarcastic, didactic, philosophical, etc.) or intention (e.g. question, complaint, request, etc.) For this project, you will consider a simple binary classification problem that detects whether the sentiment associated with text is positive or negative.

You will use the IMDB Movie Review Dataset [1] that contains 1000 training and testing examples. The model is to be built and trained on the training examples, and to be tested against the test examples.

3 Task 1: Load the data (10 points)

For this task you will load the data, create a vocabulary and encode the reviews as integers. Specifically, you will write 4 methods.

1. `read_file(path_to_dataset)`: This method takes in the path to a tar file and produces two lists, one containing movie reviews and the other containing classes.

   For example, this method outputs the lists:

   DATA = [“This movie sucks”, “Good movie”]

   LABELS = [”NEGATIVE”, ”POSITIVE”]

2. `preprocess(text)`: This method creates a vocabulary and associates the tokens with a unique integer. For the previous example, the dictionary will be:

{'this': 1, 'movie': 2, 'sucks': 3, 'good': 4}

3. `encode_review(vocab, text)`: Returns integer-encoded movie reviews. For the previous example, the reviews will now look like:

   DATA = [[1, 2, 3], [4, 2]]

4. `encode_labels(labels)`: Here you integer-encode the labels if you did not do that while reading the files. Labels will look like:

   LABELS = [0, 1]

5. `pad_zeros(encoded_reviews, seq_length)`: Here you pad zeros to each review to make all reviews have equal length. For the previous example, lets say you want seq_length to be 5. Then, the reviews will look like:

   DATA = [[1, 2, 3, 0, 0], [4, 2, 0, 0, 0]]

4 Task 2: Build the Embedding dictionary (8 points)

For this task you will load the pre-trained embedding vectors from Word2Vec. Specifically, you will associate each token id with its pre-trained vector.

For example, if the word2vec file contains these vectors:
this: 0.001 0.102 -1.221 movie: 9.011 -0.119 2.112 the: 1.223 0.911 -2.113

And the dictionary you obtained from the previous step was:
{'this': 1, 'movie': 2, 'sucks': 3, 'good': 4}

Then your embedding dict should look like:

{1: tensor[0.001 0.102 -1.221],
2: tensor[9.011 -0.119 2.112],
3: tensor[0.0 0.0 0.0],
4: tensor[0.0 0.0 0.0]}

Note that the tensors associated with words 'sucks' and 'good' are zeros because the embedding file does not have embedding vectors for these tokens. (One way to circumvent the problem is to replace the words by their lemmas or synonyms)

5 Task 3: Create a TensorDataset and DataLoader (6 points)

Look into how to create TensorDatasets and DataLoaders for effective training and testing. These classes are imported in the skeleton python file.

6 Task 4: Define the baseline model (12 points)

Create the baseline deep learning model that has a very simple architecture. The model implements two layers:

1. Embedding layer: This takes in an input sample and represents it as the word embeddings obtained from the previous task.
2. FCN: A fully connected layer is added on top of the Embedding layer that gives you the class value associated with the input.

7 Task 5: Add an RNN layer to the baseline model (18 points)

Between the embedding layer and the FCN, add an RNN layer. Experiment with the following hyper-parameters in the RNN layer:

- Type of RNN used: Vanilla RNN vs GRU vs LSTM
- Number of RNN layers: Observe the effect of increasing number of RNN layers from 1 to 2.
- Directionality: Observe the effect of making the RNN layer bi-directional.

8 Task 6: Replace the RNN by self-attention (18 points)

Replace the RNN you implemented in the previous model by self-attention. You can read more about self-attention in [2]. Observe if the accuracy changes after you have added self-attention to the model.

NOTE: Model architectures are provided as a reference in Figure 1

9 Task 7: Train and test the model (18 points)

Look into how the model can be trained and tested. Define a suitable loss and optimizer function. Experiment with different hyper-parameters such as learning rate, number of epochs and batch size. You will use accuracy as a metric for making all comparisons.

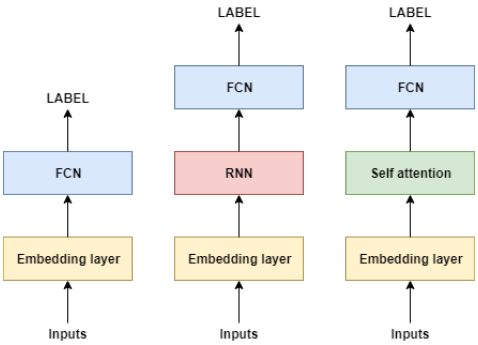

Figure 1: Simplified architectures for the three models to be implemented.

10 External links

Here are some external links for you to check out. Many of them contain pyTorch implementations of RNNs and self-attention. Try to adapt their implementations into the source code.

1. https://pytorch.org/docs/stable/index.html
2. https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948
3. https://colab.research.google.com/github/agungsantoso/deep-learning-v2-pytorch/blob/master/sentiment-rnn/Sentiment_RNN_Exercise.ipynb
4. https://towardsdatascience.com/how-to-code-the-transformer-in-pytorch-24db27c8f9ec

NOTE: All project files are available at: http://www.hlt.utdallas.edu/~takshak/project.zip

You are given the training and test data, a pre-trained word embedding file, and a skeleton python file. Note that this skeleton python file assumes you will write your code in pyTorch. If you are using Tensorflow, Caffe or any other API, please find equivalent class definition and methods.

What to submit:

1. All your source code bundled into a zip file with a README giving clear instructions on how to run the code. If you have used any external packages, please provide instructions on how to install those packages.
2. A PDF report detailing your project, and results that you obtained. This report is worth **10 points**.
3. If you are unable to complete any of the mentioned tasks, you can include some details on the things you tried out for that task. You MAY receive some partial credit for trying ☺.

References

[1] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity. In *Proceedings of ACL*, pages 271–278, 2004.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.