SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.040 Natural Language Processing, Fall 2024

Design Project

Due 13 December 2024, 23:59pm

Please start this project early.

Instructions

Please read the following instructions carefully before you start this project:

- This is a group project. You are allowed to form groups in any way you like, but each group must consist of either 2 or 3 people.
- Each group should submit code together with a report summarizing your work, and give clear instructions on how to run your code. Please also submit your system's outputs. Your output should be in the same column format as that of the training set.
- You are given **8** weeks to work on the project. Week **13** will be reserved as the "Final Project Week" for you to work on the project (i.e., there will be no class for that week). Please plan and budget your time well.
- Please use Python to complete this project.

Project Summary

Welcome to the design project for our natural language processing (NLP) course offered at SUTD! In this project, you will undertake an NLP task in *sentiment analysis*. We will begin by guiding you through data understanding, pre-processing, and instructing you to construct RNN and CNN models for the task. Afterward, you are encouraged to develop your own model to improve the results. The final test set will be released on 11 December 2024 at 5pm (48 hours before the final project deadline). You will use your own system to generate the outputs. The system with the highest F1 score will be announced as the winner for each challenge. If no clear winner emerges from the test set results, further analysis or evaluations may be conducted.

1 Task Introduction and Data pre-processing (20 points)

1.1 Sentiment Analysis

With the proliferation of online social media and review platforms, vast amounts of opinionated data have been generated, offering immense potential to support decision-making processes. Sentiment analysis examines people's sentiments expressed in text, such as product reviews, blog comments, and forum discussions. It finds wide applications in diverse fields, including politics (e.g., analyzing public sentiment

towards policies), finance (e.g., evaluating market sentiments), and marketing (e.g., product research and brand management).

Since sentiments can often be categorized into discrete polarities or scales (e.g., positive or negative), sentiment analysis can be framed as a text classification task. This task involves transforming text sequences of varying lengths into fixed-length categorical labels.

1.2 Data pre-processing

In this project, we will utilize Stanford's large movie review dataset for sentiment analysis. The dataset consists of a training set and a testing set, each containing 25,000 movie reviews sourced from IMDb. Both datasets have an equal number of "positive" and "negative" labels, representing different sentiment polarities. Please download and extract this IMDb review dataset in the path `../data/aclImdb`.

Hints: While the following instructions are based on a split of 25,000 for training and 25,000 for testing, you are free to choose your own dataset split, as we will provide a separate test set for the final evaluation. However, any changes you make to the default split must be clearly indicated in your report. Failure to explicitly mention such modifications may result in a penalty.

Question 1 [code] (5 points)

Complete the function `read_imdb`, which reads the IMDb review dataset text sequences and labels. Then, run the sanity check cell to check your implementation.

Question 2 [code] (5 points)

Treating each word as a token and filtering out words that appear less than 5 times, we create a vocabulary out of the training dataset. After tokenization, please plot the histogram of review lengths in tokens. (Hint: you can use *matplotlib* package to draw a histogram.) Then, run the sanity check cell to check your implementation.

Question 3 [code] (5 points)

Create data iterator `train_iter`, at each iteration, a minibatch of examples are returned. Let's set the mini-batch size to 64.

Question 4 [code] (5 points)

Finally, wrap up the above steps into the `load_data_imdb` function. It returns training and test data iterators and the vocabulary of the IMDb review dataset.

2 Using RNN for sentiment analysis (30 points)

Similar to word similarity and analogy tasks, pre-trained word vectors can also be applied to sentiment analysis. Given that the IMDb review dataset is relatively small, using text representations pre-trained on large-scale corpora can help mitigate model overfitting. Each token can be represented using the pre-trained GloVe model, and these token embeddings can be fed into a multilayer bidirectional RNN to generate a sequence representation of the text, which will then be transformed into sentiment analysis outputs. Later, we will explore an alternative architectural approach for the same downstream task.

Question 5 [code] (10 points)

In text classification tasks, such as sentiment analysis, a varying-length text sequence is transformed into fixed-length categorical labels. Following the instructions, please complete `BiRNN` class, each token in a text sequence receives its individual pre-trained GloVe representation through the embedding layer (`self.embedding`). The entire sequence is then encoded by a bidirectional RNN (`self.encoder`). Specifically, the hidden states from the last layer of the bidirectional LSTM, at both the initial and final time steps, are concatenated to form the representation of the text sequence. This representation is subsequently passed through a fully connected layer (`self.decoder`) to produce the final output categories, which in this case are "positive" and "negative".

Question 6 [code] (10 points)

After loading the pretrained word vectors, we can now start to train the model. Please use `Adam` optimizer and `CrossEntropyLoss` for training and draw a graph about your training loss, training acc and testing acc.

Question 7 [code] (10 points)

Once you have completed the training, it's time to evaluate your model's performance. Implement the function `predict_sentiment` to predict the sentiment of a text sequence using the trained model `net`. Next, define the function `cal_metrics` to assess your model on the test set by calculating both accuracy and the F1-score, including precision and recall. Finally, print out the evaluation results. Save the prediction results for test samples and submit it in the zip file.

3 Using CNN for sentiment analysis (20 points)

Although CNNs were originally designed for computer vision, they have been widely adopted in natural language processing as well. Conceptually, a text sequence can be viewed as a one-dimensional image, allowing one-dimensional CNNs to capture local features, such as n-grams, within the text. We will use the textCNN model to demonstrate how to design a CNN architecture for representing single text.

Using one-dimensional convolution and max-over-time pooling, the textCNN model takes individual pre-trained token representations as input, then extracts and transforms these sequence representations for downstream tasks.

For a single text sequence with n tokens represented by d-dimensional vectors, the width, height, and number of channels of the input tensor are n, 1, and d, respectively. The textCNN model processes the input as follows:

1. Define multiple one-dimensional convolutional kernels and apply convolution operations on the inputs. Convolution kernels with varying widths capture local features across different numbers of adjacent tokens.

2. Apply max-over-time pooling to all output channels, then concatenate the resulting scalar outputs into a vector.

3. Pass the concatenated vector through a fully connected layer to generate the output categories. Dropout can be applied to reduce overfitting.

Question 8 [code] (10 points)

Implement the `textCNN` model class. Compared with the bidirectional RNN model in Section 2, besides replacing recurrent layers with convolutional layers, we also use two embedding layers: one with trainable weights and the other with fixed weights.

Question 9 [code] (10 points)

Similar to what we have done in Section 2 with RNN, train the CNN model with the same optimizer and loss function. Draw a graph about your training loss, training acc and testing acc. Use the prediction function you defined in Question 7 to evaluate your model performance.

4 Design Challenge (30 points)

Now it's time to come up with your own model! You are free to decide what model you want to choose or what architecture you think is better for this task. You are also free to set all the hyperparameter for training (do consider the overfitting issue and the computing cost). From here we will see how important choosing/designing a better model architecture could be when building an NLP application in practice (however, there is no requirement for you to include such comparisons in your report).

You are allowed to use external packages for this challenge, but we require that you fully understand the methods/techniques that you used, and you need to clearly explain such details in the submitted report. We will evaluate your system's performance on the held-out test set, which will only be released 48 hours before the deadline. Use your new system to generate the outputs. The system that achieves the highest F1 score will be announced as the winner for the challenge. We may perform further analysis/evaluations in case there is no clear winner based on the released test set.

Let's summarize this challenge:

[Model] You are required to develop your own model for sentiment analysis, with no restrictions on the model architecture. You may choose to follow RNN/CNN structures or experiment with Transformer-based models. In your submitted report, you must provide a detailed explanation of your model along with the accompanying code. Additionally, you are required to submit your model so that we can reproduce your results.

(10 points)

[Evaluation] For a fair comparison, you must train your new model on the same dataset provided to you. After training, evaluate your model on the test set. You are required to report the *precision*, *recall*, *F1 scores*, and accuracy of your new model. Save the predicted outcome for the test set and include it in the submission.

Hint: You will be competing with other groups on the same test set. Groups with higher performance will receive more points. For instance, if your group ranks 1st among all groups, you will receive 15 points for this section.

(15 points)

[Report] You are required to submit a report for your model. The report must include, at a minimum, the following sections: Model Description, Training Settings (e.g., dataset, hyperparameters), Performance, Code to run the model, and a breakdown of how the work was divided among team members. You

are encouraged to include any additional details you deem important. Instructions on how to run the code can either be included in a separate README file or integrated into the report, as long as they are clearly presented.

Please provide a thorough explanation of the model or method you used for designing the new system, along with clear documentation on how to execute the code. We will review your code, and may request an interview if we have any questions about it.

Note: Reports, code, and README files that are of low quality or poorly written will be penalized, so please ensure they are well-organized and clearly formatted. If we are unable to reproduce your model or run your code, you will not receive any points for this challenge.

(5 points)

Items To Be Submitted

Upload to eDimension a single ZIP file containing the following: (Please make sure you have only one submission from each team.)

- Solution for final_project.ipynb, you should convert it to PDF like the homeworks.
- A report detailing the approaches and results. Make sure you have include your group information in the report.
- Source code (.py files) with README (instructions on how to run the code)
- Output files
  - `Your Model in Design Challenge (You do not need to submit your RNN/CNN model in Section 2&3)`
  - `Prediction Output of RNN`
  - `Prediction Output of CNN`
  - `Prediction Output of Your Own Model`