SC4002 / CE4045 / CZ4045 Natural Language Processing

AY 2024-2025 Assignment

Instruction

- This is a group assignment. Each group consists of 5 to 6 students.
- The assignment constitutes **35%** of your total grade for this course.
- Please submit your assignment solution via NTULearn by **Sunday, 10th November at 11:59pm SGT**. For late submission, there will be 5% point deduction each calendar day after the deadline. You are advised to submit not more than three times to the system, and only the last submission will be graded and time-stamped.
- Please name your file under this format "SC4002_X" and replace "X" with your assigned group ID (e.g., "SC4002_G6" if your group ID is G6).
- Each group should submit (1) a single report in PDF which contains the complete write-up describing your design and answers; (2) a README.txt which gives instructions to run the code and explanations of sample output obtained from your code; (3) a zip file containing all your source code (in python).
- Please list all the full names of the group members and your group ID in the cover page of the report. All members in the same group will receive the same grade. However, contributions of each individual member to the assignment should be clearly indicated in the cover page of the report.
- Keep your report within 10 pages (excluding the cover page).

Problem

In this assignment, you will build a general classification system built on top of the existing pretrained word embeddings, e.g., **word2vec** or **Glove**. The system is used to perform sentence classification tasks, specifically **Sentiment Classification** in this project, which assigns a sentiment label to each sentence. The objective of this exercise is for you to be familiar with typical NLP applications of word embeddings and typical deep learning models for sentence classification tasks. You will also practice how to train effective classifiers from pretrained word embeddings and evaluate the performances.

Part 0. Dataset Preparation

We will be using the movie review dataset introduced in https://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.README.1.0.txt. To load this dataset, you need to install the "datasets" library via `pip install datasets`. Then you can use the following code snippet:

```
from datasets import load_dataset
dataset = load_dataset("rotten_tomatoes")
train_dataset = dataset['train']
validation_dataset = dataset['validation']
test_dataset = dataset['test']
```

Using the original train-valid-test split provided in the above code, you will perform model training on the training dataset, configure your model (e.g., learning rate, batch size, number of training epochs) on the validation dataset, and conduct evaluation on the test dataset.

Part 1. Preparing Word Embeddings

As the first step of building your model, you need to prepare the word embeddings to form the input layer of your model. You are required to choose only from **Word2vec** or **Glove** to initialize your word embedding matrix. The word embedding matrix stores the pre-trained word vectors (taken from Word2vec or Glove) where each row corresponds to a vector for a specific word in the vocabulary formed from your task dataset.

Question 1. Word Embedding

(a) What is the size of the vocabulary formed from your training data?

(b) We use **OOV** (out-of-vocabulary) to refer to those words appeared in the training data but not in the Word2vec (or Glove) dictionary. How many OOV words exist in your training data?

(c) The existence of the OOV words is one of the well-known limitations of Word2vec (or Glove). Without using any transformer-based language models (e.g., BERT, GPT, T5), what do you think is the best strategy to mitigate such limitation? Implement your solution in your source code. Show the corresponding code snippet.

Part 2. Model Training & Evaluation - RNN

Now with the pretrained word embeddings acquired from Part 1 and the dataset acquired from Part 0, you need to train a deep learning model for sentiment classification using the training set, conforming to these requirements:

- Use the pretrained word embeddings from Part 1 as inputs; do not update them during training (they are "frozen").
- Design a simple recurrent neural network (RNN), taking the input word embeddings, and predicting a sentiment label for each sentence. To do that, you need to consider how to aggregate the word representations to represent a sentence.
- Use the validation set to gauge the performance of the model for each epoch during training. You are required to use **accuracy** as the performance metric during validation and evaluation.
- Use the mini-batch strategy during training. You may choose any preferred optimizer (e.g., SGD, Adagrad, Adam, RMSprop). Be careful when you choose your initial learning rate and mini-batch size. (You should use the validation set to determine the optimal configuration.) Train the model until the **accuracy** score on the validation set is not increasing for a few epochs.
- Evaluate your trained model on the test dataset, observing the **accuracy** score.

Question 2. RNN

(a) Report the final configuration of your best model, namely the number of training epochs, learning rate, optimizer, batch size.

(b) Report the **accuracy** score on the test set, as well as the **accuracy** score on the validation set for each epoch during training.

(c) RNNs produce a hidden vector for each word, instead of the entire sentence. Which methods have you tried in deriving the final sentence representation to perform sentiment classification? Describe all the strategies you have implemented, together with their accuracy scores on the test set.

Part 3. Enhancement

The RNN model used in Part 2 is a basic model to perform the task of sentiment classification. In this section, you will design strategies to improve upon the previous model you have built. You are required to implement the following adjustments:

1. Instead of keeping the word embeddings fixed, now update the word embeddings (the same way as model parameters) during the training process.
2. As discussed in Question 1(c), apply your solution in mitigating the influence of OOV words and train your model again.
3. Keeping the above two adjustments, replace your simple RNN model in Part 2 with a biLSTM model and a biGRU model, incorporating recurrent computations in both directions and stacking multiple layers if possible.
4. Keeping the above two adjustments, replace your simple RNN model in Part 2 with a Convolutional Neural Network (CNN) to produce sentence representations and perform sentiment classification.
5. Further improve your model. You are free to use any strategy other than the above mentioned solutions. Changing hyper-parameters or stacking more layers is not counted towards a meaningful improvement.

Question 3. Enhancement

(a) Report the accuracy score on the test set when the word embeddings are updated (Part 3.1).

(b) Report the accuracy score on the test set when applying your method to deal with OOV words in Part 3.2.

(c) Report the accuracy scores of biLSTM and biGRU on the test set (Part 3.3).

(d) Report the accuracy scores of CNN on the test set (Part 3.4).

(e) Describe your final improvement strategy in Part 3.5. Report the accuracy on the test set using your improved model.

(f) Compare the results across different solutions above and describe your observations with possible discussions.