SC4002 / CE4045 / CZ4045 Natural Language Processing

AY 2025-2026 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 **Wednesday, 12th 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 zip file containing all your source codes (**in ipython notebook**). Make sure your ipython notebook contains all the output logs you have produced in your experiments.
- 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 **Topic Classification** in this project, which assigns a topic 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 TREC question dataset introduced in https://cogcomp.seas.upenn.edu/Data/QA/QC/, a collection of questions designed to be classified into a small set of categories. To load this dataset, you need to install the "torchtext" library via `pip install torchtext==0.4.0`. Note that version 0.4.0 needs to be used to avoid inconsistencies. Then you can use the following code snippet:

```python
from torchtext import data, datasets
import torch
# For tokenization
TEXT = data.Field(tokenize='spacy',
                  tokenizer_language='en_core_web_sm',
                  include_lengths=True)
# For multi-class classification labels
LABEL = data.LabelField()
# Load the TREC dataset
train_data, test_data = datasets.TREC.splits(TEXT, LABEL, fine_grained=False)
```

Each data instance can be printed via

```
print(vars(train_data.examples[0]))
```

Alternatively, you can also download the dataset from the original website https://cogcomp.seas.upenn.edu/Data/QA/QC/. Make sure you download "**Training set 5(5500 labeled questions)**" for training and "**Test set: TREC 10 questions**" for testing. Before working on the model and training, you need to split the training data into train and validation sets (with ratio 0.8:0.2). Subsequently, you can perform model training on the train set, 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 using the pre-trained vectors 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 according to your tokenization method?

(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? What is the number of OOV words for each topic category?

(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.

(d) Select the 20 most frequent words from each topic category in the training set (removing stopwords if necessary). Retrieve their pretrained embeddings (from Word2Vec or GloVe). Project these embeddings into 2D space (using e.g., t-SNE or Principal Component Analysis). Plot the points in a scatter plot, color-coded by their topic category. Attach your plot here. Analyze your findings.

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 topic classification using the training set, conforming to these requirements:

- Use the pretrained word embeddings from Part 1 as inputs, together with your implementation in mitigating the influence of OOV words; make them learnable parameters during training (they are updated).
- Design a simple recurrent neural network (RNN), taking the input word embeddings, and predicting a topic 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.
- Try different regularization techniques to mitigate overfitting.
- 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 and hidden dimension.

(b) Report all the regularization strategies you have tried. Compare the accuracy on the test set among all strategies and the one without any regularization.

(c) For the best configuration and regularization strategy in your experiments, plot the training loss curve and validation accuracy curve during training with x-axis being the number of training epochs. Discuss what the curves inform about the training dynamics.

(d) 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.

(e) Report topic-wise accuracy (accuracy for each topic) on the test set for the best model you have. Discuss what may cause the difference in accuracies across different topic categories.

Part 3. Enhancement

The RNN model used in Part 2 is a basic model to perform the task of topic 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. 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.
2. Replace your simple RNN model in Part 2 with a Convolutional Neural Network (CNN) to produce sentence representations and perform topic classification.
3. 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.
4. Instead of looking at generic performance improvement, think about targeted improvement on specific topics. Based on your findings in Part 2(e), design strategies aimed at improving performance specifically for those weaker topics.

Question 3. Enhancement

(a) Plot the training loss curve and validation accuracy curve of biLSTM and biGRU. Report their accuracies on the test set (Part 3.1).

(b) Plot the training loss curve and validation accuracy curve of CNN. Report its accuracy score on the test set (Part 3.2).

(c) Describe your final improvement strategy in Part 3.3. Plot the corresponding training loss curve and validation accuracy curve. Report the accuracy on the test set.

(d) Describe your strategy for improvement of weak topics in Part 3.4. Report the topic-wise accuracy on the test set after applying the strategy and compare with the results in Part 2(e).