SC4002 / CE4045 / CZ4045 Natural Language Processing

AY 2023-2024 Assignment

Instruction

- This is a group assignment. Each group consists of 4 to 5 students.
- The assignment constitutes **35%** of your total grade for this course.
- Please submit your assignment solution via NTULearn by **Friday, 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.
- Please list all the full names of the group members in the first 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 first page of the report.

Problem

In this assignment, you will build a general classification system built on top of the existing pretrained word embeddings: **word2vec**. The system should be capable of (1) conducting a sequence tagging task: **Named Entity Recognition (NER)** which classifies each word to a specific label from a pre-defined label set; (2) performing a sentence classification task: **Question Classification** which assigns each question to a specific topic category. The objective of this exercise is for you to be familiar with typical NLP applications of word embeddings and be able to distinguish the model design for sequence tagging from that of sentence classification. You will also practice how to train effective classifiers from pretrained word embeddings and evaluate the performances.

Part 1. Sequence Tagging: NER

Build an NER classifier using pretrained word embeddings. Specifically, NER aims to identify named entities (words or phrases) from each sentence given. To do that, we can formulate this task to a sequence tagging problem where each word in the sentence will be assigned a label. You can use the IO, BIO or any other sequence tagging scheme of your choice.

1.1 Word Embedding

As the first step, you need to download **word2vec** embeddings. You can follow these steps:

1. Word2vec could be accessed via the gensim library (with installation guide).
2. Download the pretrained word2vec embeddings following the instructions under Section Pretrained Models. Download the embeddings under this name: **word2vec-google-news-300**.

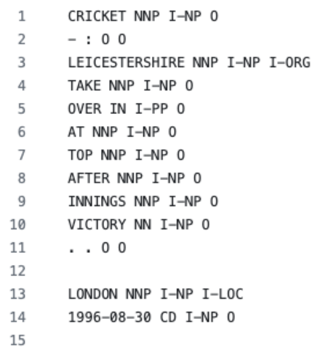
3. Now you should be able to query the vector of any word by specifying the word as the key. For example, if you store the pretrained embeddings under variable name *w2v*, you can run *w2v*['computer'] to retrieve the vector for 'computer'.

Question 1.1

Based on **word2vec** embeddings you have downloaded, use cosine similarity to find the most similar word to each of these words: (a) "student"; (b) "Apple"; (c) "apple". Report the most similar word and its cosine similarity.

1.2 Data

Then you can start to prepare the dataset. For NER, you will work with CoNLL2003 (click and download "eng.testa", "eng.testb", "eng.train"). Before training, you need to preprocess the dataset such that each of them contains a training file, a development file and a test file. The development file is used to select the best model during training. The test file is used for final evaluation. For CoNLL2003, the training, development and test file corresponds to "eng.train", "eng.testa" and "eng.testb", respectively. Note that you only need to use the first and the last column of each line corresponding to the input word and the word label, respectively. A screenshot of the data is shown in Figure 1. In this example, there are two sentences (separated by '\n'). The input for each sentence is composed of the words from the first column. For example, the first sentence in Figure 1 corresponds to "*CRICKET - LEICESTERSHIRE TAKE OVER AT TOP AFTER INNINGS VICTORY.*" The label for each of the first 3 words *CRICKET*, -, *LERCESTERSHIRE* corresponds to 'O', 'O', 'I-ORG', respectively.



```
CRICKET NNP I-NP O
- : O O
LEICESTERSHIRE NNP I-NP I-ORG
TAKE NNP I-NP O
OVER IN I-PP O
AT NNP I-NP O
TOP NNP I-NP O
AFTER NNP I-NP O
INNINGS NNP I-NP O
VICTORY NN I-NP O
. . O O

LONDON NNP I-NP I-LOC
1996-08-30 CD I-NP O
```

Figure 1: An example of raw data from CoNLL2003.

Question 1.2

(a) Describe the size (number of sentences) of the training, development and test file for CoNLL2003. Specify the complete set of all possible word labels based on the tagging scheme (IO, BIO, etc.) you chose.

(b) Choose an example sentence from the training set of CoNLL2003 that has at least two named entities with more than one word. Explain how to form complete named entities from the label for each word, and list all the named entities in this sentence.

1.3 Model

Now with the pretrained word embeddings acquired from Section 1.1, and the CoNLL2003 dataset acquired from Section 1.2, you need to train an NER model using the training set, conforming to these requirements:

- Use the pretrained word embeddings from Section 1.1 as inputs; do not update them during training (they are "frozen").
- Design a neural network transforming the input for each word to its final vector representation, which will be fed into the softmax classifier to predict the final label for each word. The neural network could be a simple linear layer, a feedforward network (a linear transformation plus a nonlinear activation function), or a recurrent neural network (RNN/LSTM). You are encouraged to use more effective networks (e.g., LSTM) because the performance of the model will be taken into consideration when graded.

- Use the development set to evaluate the performance of the model for each epoch during training. Please use **f1_score** to measure the performance. For evaluation metric and code, refer to the following link: `https://github.com/chakki-works/seqeval/tree/master`. (Please make sure your sequence tagging scheme aligns with the provided evaluation code. Otherwise, you may revise the code accordingly.)

- 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 may use the development set to check the performance and decide the optimal configuration.) Train the network until the **f1_score** on the development set is not increasing. Use the trained network to classify words in the test set.

Question 1.3

(a) Discuss how you deal with new words in the training set which are not found in the pretrained dictionary. Likewise, how do you deal with new words in the test set which are not found in either the pretrained dictionary or the training set? Show the corresponding code snippet.

(b) Describe what neural network you used to produce the final vector representation of each word and what are the mathematical functions used for the forward computation (i.e., from the pretrained word vectors to the final label of each word). Give the detailed setting of the network including which parameters are being updated, what are their sizes, and what is the length of the final vector representation of each word to be fed to the softmax classifier.

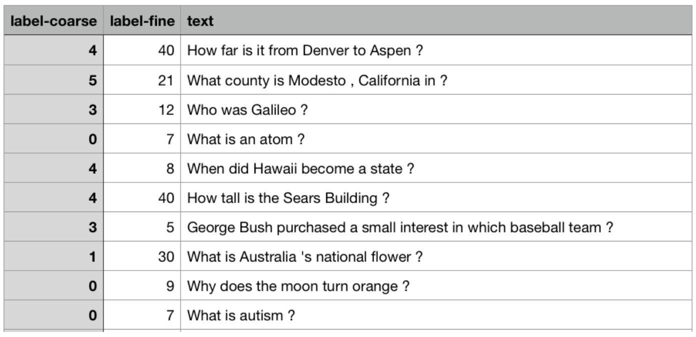
(c) Report how many epochs you used for training, as well as the running time.

(d) Report the **f1_score** on the test set, as well as the **f1_score** on the development set for each epoch during training.

Part 2. Sentence-Level Categorization: Question Classification

Now let's practice how to do sentence-level classification using the task of **Question Classification**. Specifically, Question Classification aims to categorize each question into one of the topics from a pre-defined label set. Different from the sequence tagging problem where each word in the sentence will be assigned a label, this task requires to assign a single label for the entire sentence. To do that, you need to consider how to aggregate the word representations to represent a sentence. Some possible aggregation methods include averaging over word representations, doing max pooling, taking representation of the last word if RNN is used.

For Question Classification, you will work with TREC dataset (click to download). Please ignore fine-grained labels under 'label-fine' and only use 'label-coarse' to train the model, as shown in Figure 2.



label-coarse	label-fine	text
4	40	How far is it from Denver to Aspen ?
5	21	What county is Modesto , California in ?
3	12	Who was Galileo ?
0	7	What is an atom ?
4	8	When did Hawaii become a state ?
4	40	How tall is the Sears Building ?
3	5	George Bush purchased a small interest in which baseball team ?
1	30	What is Australia 's national flower ?
0	9	Why does the moon turn orange ?
0	7	What is autism ?

Figure 2: An example of raw data from TREC.

Similar to CoNLL2003, you need to preprocess the dataset such that each of them contains a training file, a development file and a test file. For TREC, there is no development file. You are expected to form a development set from a random subset (containing 500 examples) within the original training data. Make sure you will remove these examples from the original training file when training your model. Again, you are expected to use **word2vec** as the pretrained word embeddings, following the same procedure as shown in Section 1.1.

Now train a question classification model using the training set, conforming to these requirements:

- Randomly select 4 classes from the 6 different coarse labels, and combine the remaining 2 labels to form a single class: ‘OTHERS’. Adjust the original data such that the label for each sentence is updated according to the new setting (5 classes in total). Make sure all subsequent experiments are conducted over the updated dataset.
- Use the pretrained word embeddings from Section 1.1 as inputs; do not update them during training (they are “frozen”).
- Design a neural network transforming the input for each word to its final vector representation, which will be aggergated and fed into the softmax classifier to predict the final label for each question. The neural network could be a simple linear layer, a feedforward network (a linear transformation plus a nonlinear activation function), or a recurrent neural network (RNN/LSTM). You are encouraged to use more effective networks because the performance of the model will be taken into consideration when graded.
- Use the development set to evaluate the performance of the model for each epoch during training. Please use **accuracy** to measure the performance.
- 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 may use the development set to check the performance and decide the optimal configuration.) Train the network until the **accuracy** on the development set is not increasing. Use the trained network to classify questions in the test set.

Question 2

(a) Specify all the 5 classes you used after converting from the original label set to the new setting.

(b) Describe what aggregation methods you have tried and which is finally adopted (and why). Explain the detailed function of the aggregation method you used. If you have tested different aggregation methods, list their accuracy results to support your claim.

(c) Describe what neural network you used to produce the final vector representation of each word and what are the mathematical functions used for the forward computation (i.e., from the pretrained word vectors to the final label of each word). Give the detailed setting of the network including which parameters are being updated, what are their sizes, and what is the length of the final vector representation of each word to be fed to the softmax classifier.

(d) Report how many epochs you used for training, as well as the running time.

(e) Report the accuracy on the test set, as well as the accuracy on the development set for each epoch during training.