Sentence-level Sentiment Classification with PyTorch

Homework 5 for Introduction to Deep Learning, Fall 2021

Deadline: 2020.12.13 23:59:59

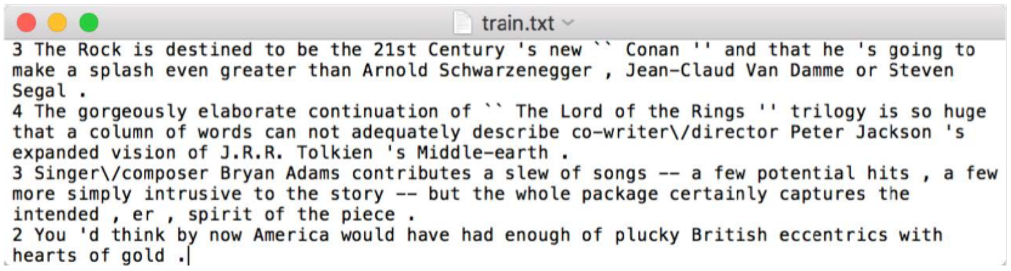
1 Introduction

Stanford Sentiment Treebank (SST) is dataset for sentiment classification in machine learning field. It contains 11855 sentences, and has been split into the training / validation / test parts, respectively containing 8,544 / 1,101 / 2,210 sentences.

Note: During training, information about testing examples should never be used in any form.

1.1 Data Format

Every line in SST: Label(Sentiment) + Data(Sentence) There are five kinds of annotations in label: 0-"very negative"; 1-"negative"; 2-"neutral" 3-"positive"; 4-"very positive". Digits in MNIST range from 0 to 9. Some examples are shown below.



```
3 The Rock is destined to be the 21st Century 's new `` Conan '' and that he 's going to
make a splash even greater than Arnold Schwarzenegger , Jean-Claud Van Damme or Steven
Segal .
4 The gorgeously elaborate continuation of `` The Lord of the Rings '' trilogy is so huge
that a column of words can not adequately describe co-writer\/director Peter Jackson 's
expanded vision of J.R.R. Tolkien 's Middle-earth .
3 Singer\/composer Bryan Adams contributes a slew of songs -- a few potential hits , a few
more simply intrusive to the story -- but the whole package certainly captures the
intended , er , spirit of the piece .
2 You 'd think by now America would have had enough of plucky British eccentrics with
hearts of gold .
```

Figure 1: Examples of data in SST.

1.2 Data Preprocessing

Torchtext is recommended for loading and preprocessing SST data. To install torchtext, you can use **pip install torchtext**.

We provide some start codes for SST DataLoader, which are included in **tips_code.py**.

To learn more about Torchtext, you can read some documents about TorchText: Torchtext Doc and SST Dataset Source Code.

1.3 Introduction for Word Embedding

Word Embedding is used in our Dataloader code. The embedding layer is used to transform the word into a dense embedding vector. This embedding layer is simply a single fully connected layer. You can see torch.nn.Embedding to learn more details. The input is firstly passed through the embedding layer to get embedded, which gives us a dense vector representation of our sentences. embedded is then fed into the RNN. For simplicity, we use pre-trained word embeddings. Codes for pre-trained embeddings are provided. You can also use other pre-trained embeddings in this task. Figure 2 shows the basic process for sentiment classification.

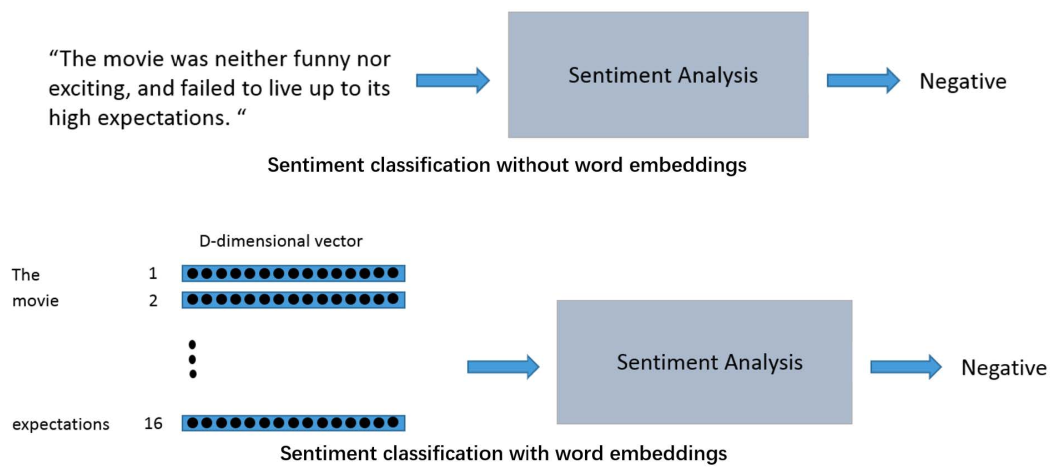

"The movie was neither funny nor exciting, and failed to live up to its high expectations. "
Sentiment Analysis
Negative
Sentiment classification without word embeddings
D-dimensional vector
The 1
movie 2
expectations 16
Sentiment Analysis
Negative
Sentiment classification with word embeddings

Figure 2: Basic process for sentiment classification

1.4 Introduction for RNN in Pytorch

RNN is a basic network for sequence processing. Pytorch provides many kinds of RNN such as "RNN", "LSTM" and "GRU". You can check them in https://pytorch.org/docs/stable/nn.html#recurrent-layers

Here are some examples: https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html#sphx-glr-beginner-nlp-sequence-models-tutorial-py

1.4.1 Example Architecture for Sentiment Classification

Here are two examples of network architecture: Figure 3 and Figure 4.

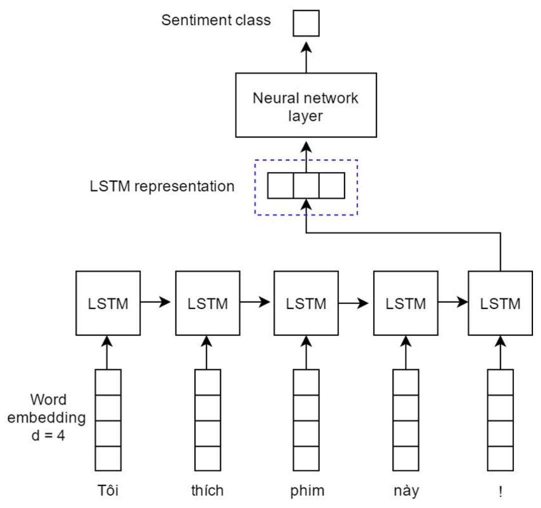
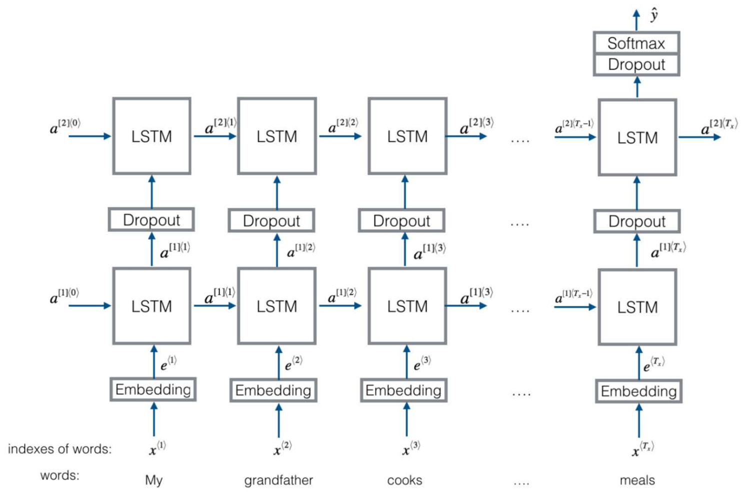


Figure 3: Model architecture example 1

RNN-type architecture is required for this homework. You should first design a model with RNN cells. We also encourage you to try other model architectures, but there will be no additional score rewards.



$\hat{y}$
Softmax
Dropout
$a^{[2]\langle 0 \rangle}$
LSTM
$a^{[2]\langle 1 \rangle}$
$a^{[2]\langle 2 \rangle}$
$a^{[2]\langle 3 \rangle}$
$a^{[2]\langle T_x-1 \rangle}$
$a^{[2]\langle T_x \rangle}$
Dropout
$a^{[1]\langle 1 \rangle}$
$a^{[1]\langle 2 \rangle}$
$a^{[1]\langle 3 \rangle}$
$a^{[1]\langle T_x \rangle}$
$a^{[1]\langle 0 \rangle}$
$a^{[1]\langle T_x-1 \rangle}$
$e^{\langle 1 \rangle}$
$e^{\langle 2 \rangle}$
$e^{\langle 3 \rangle}$
$e^{\langle T_x \rangle}$
Embedding
indexes of words:
$x^{\langle 1 \rangle}$
$x^{\langle 2 \rangle}$
$x^{\langle 3 \rangle}$
$x^{\langle T_x \rangle}$
words:
My
grandfather
cooks
meals

Figure 4: Model architecture example 2

2 Requirements

You are required to implement Sentence-level Sentiment Classification with PyTorch. There are no implementation limits. All parts of implementation depend on you. (e.g. types of rnn, number of layers/units, loss, optimizer...) You are encouraged to use techniques such as bidirectional, dropout and attention, to improve the accuracy. **You need to submit all codes and a report** with the following requirements:

- Illustrate your network architecture with words and figures in your report.
- Show your best results in your report. (This is a must)
- (Some suggestion to enrich your report) Show your hyper-parameters, plot the training loss curve, plot validation accuracy curve in the report.

3 Attention

- You need to submit all codes and a report (at least two pages **in PDF format**).
- Do not paste a lot of codes in your report.
- **Plagiarism (from the internet) is not permitted.**