Sentence-level Sentiment Classification (score 10)

Homework 4 for Deep Learning, Autumn 2023

Deadline: 2023.12.11 12:00

Attention

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

1 Objective Questions (Score 2)

These questions are all multiple choice questions. One or more options may be correct.

1.1 (Score 0.5)

Which models can keep longer short-term memory than the Elman network?

A. Jordan network

B. LSTM

C. GRU

1.2 (Score 0.5)

In GRU:

$$\boldsymbol{h}^{(t)} = \boldsymbol{z}^{(t)} \odot \boldsymbol{h}^{(t-1)} + (1 - \boldsymbol{z}^{(t)}) \odot \tilde{\boldsymbol{h}}^{(t)}$$
$$\tilde{\boldsymbol{h}}^{(t)} = \sigma_h(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h(\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}) + \boldsymbol{b}_h)$$

, What's the ideal case for keeping the memory $\boldsymbol{h}^{(n)}$ obtained at $t = n$ forever?

A. $\boldsymbol{z}^{(t)} = 0, \boldsymbol{r}^{(t)} = 0, \forall t \geq n + 1$

B. $\boldsymbol{z}^{(t)} = 0, \boldsymbol{r}^{(t)} = 1, \forall t \geq n + 1$

C. $\boldsymbol{z}^{(t)} = 1, \forall t \geq n + 1$

1.3 (Score 0.5)

Which models can be trained using the teacher forcing method?

A. Elman Network

B. Jordan Network

C. LSTM

D. GRU

1.4 (Score 0.5)

Which of the following is NOT an advantage of GRU over LSTM?

A. GRU performs better on tasks with long-term dependencies.

B. GRU has fewer parameters than LSTM, which can make it more efficient to train.

C. GRU is simpler than LSTM.

D. GRU can be a better choice when computational resources are limited.

2 Programming Practice (Score 8)

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, respectivelycontaining 8,544 / 1,101 / 2,210 sentences.

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

In this homework, you are required to implement a **RNN-type neural network** to perform Sentence-level Sentiment Classification. 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 should use **pytorch, mindspore or jittor** framework.

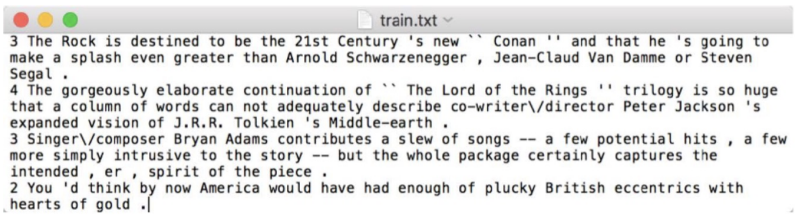
2.1 Dataset Introduction

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.

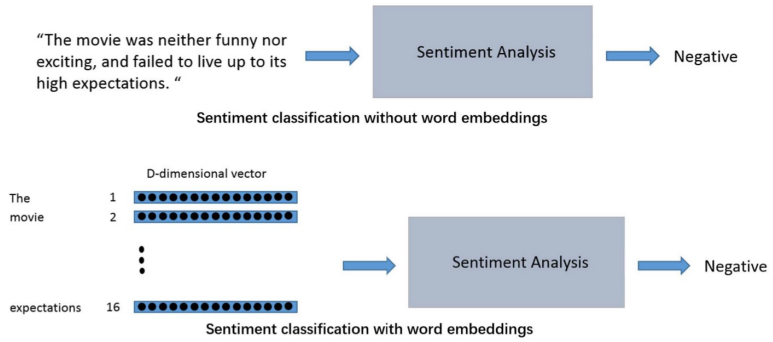
To learn more about Torchtext, you can read some documents about TorchText: https://torchtext.readthedocs.io/en/latest/datasets.html#sst .

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”. Some examples are shown below.



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 below shows the basic process for sentiment classification.

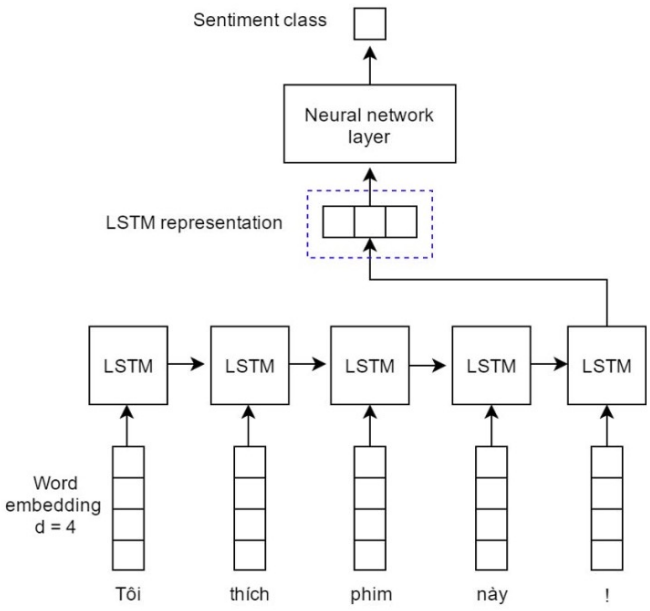


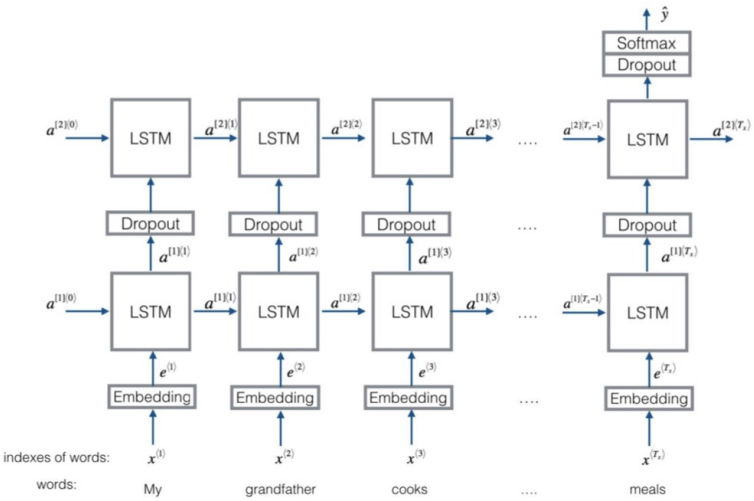
2.2 RNN-type Neural Network Introduction

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

Here are two examples of network architecture.




Softmax
Dropout
LSTM
Dropout
Embedding
indexes of words:
words:
My
grandfather
cooks
meals