Sentence-level Sentiment Classification with PyTorch

Homework-5

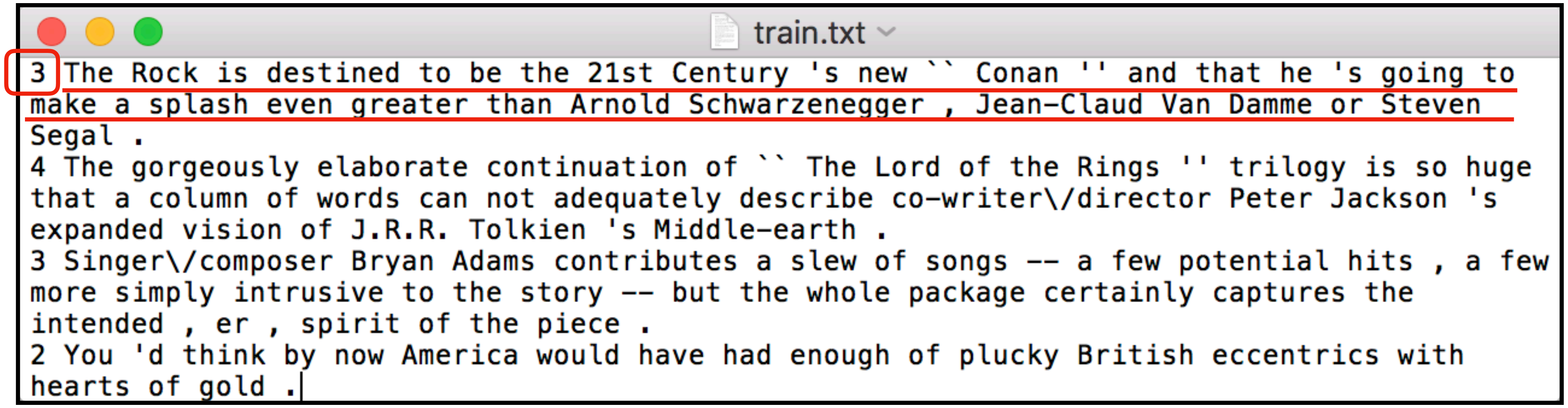
Deadline: 2018.12.16 23:59:59

SST Dataset

- Stanford Sentiment Treebank (SST) dataset contains 11,855 sentences, and has been split into the training / validation / test parts, respectively containing 8,544 / 1,101 / 2,210 sentences.

- Every line: Label(Sentiment) + Data(Sentence)

    - 0: very negative; 1: negative; 2: neutral
      3: positive; 4: very positive.



```
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 .
```

TorchText

- The torchtext package consists of data processing utilities and popular datasets for natural language.
    - pip install torchtext
    - conda install nltk
- TorchText provides the SST dataset.
- Please read some documents about TorchText: [TorchText doc], [SST Dataset Code], [Using SST], [zhihu]
- **We provide some start codes for SST DataLoader.**

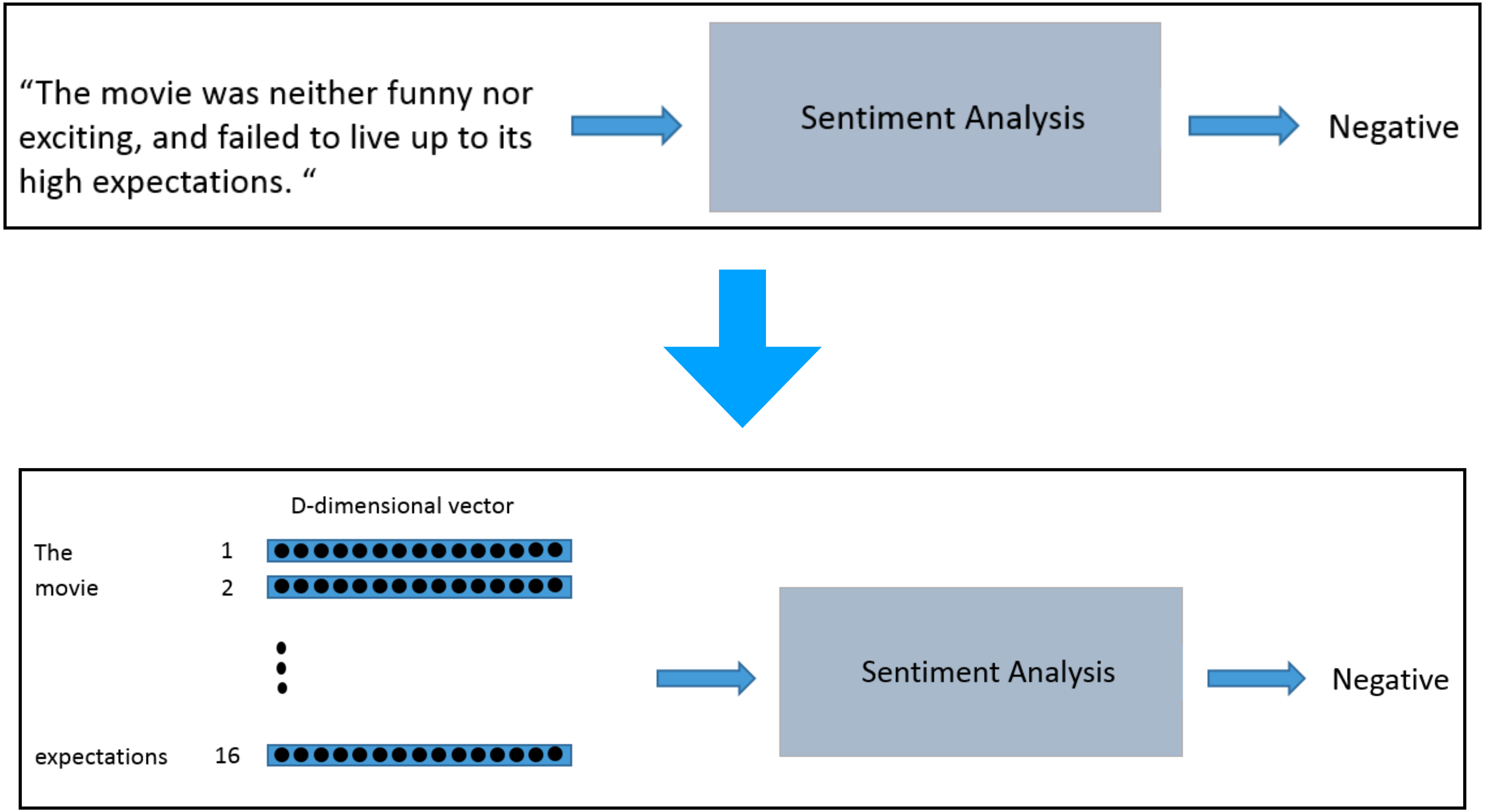
Word Embedding

- The embedding layer is used to transform the word into a dense embedding vector. This embedding layer is simply a single fully connected layer. See torch.nn.Embedding.

- 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.

Word Embedding



"The movie was neither funny nor exciting, and failed to live up to its high expectations. "

Sentiment Analysis

Negative

D-dimensional vector

The	1	●●●●●●●●●●●●●●●
movie	2	●●●●●●●●●●●●●●●
	⋮	
expectations	16	●●●●●●●●●●●●●●●

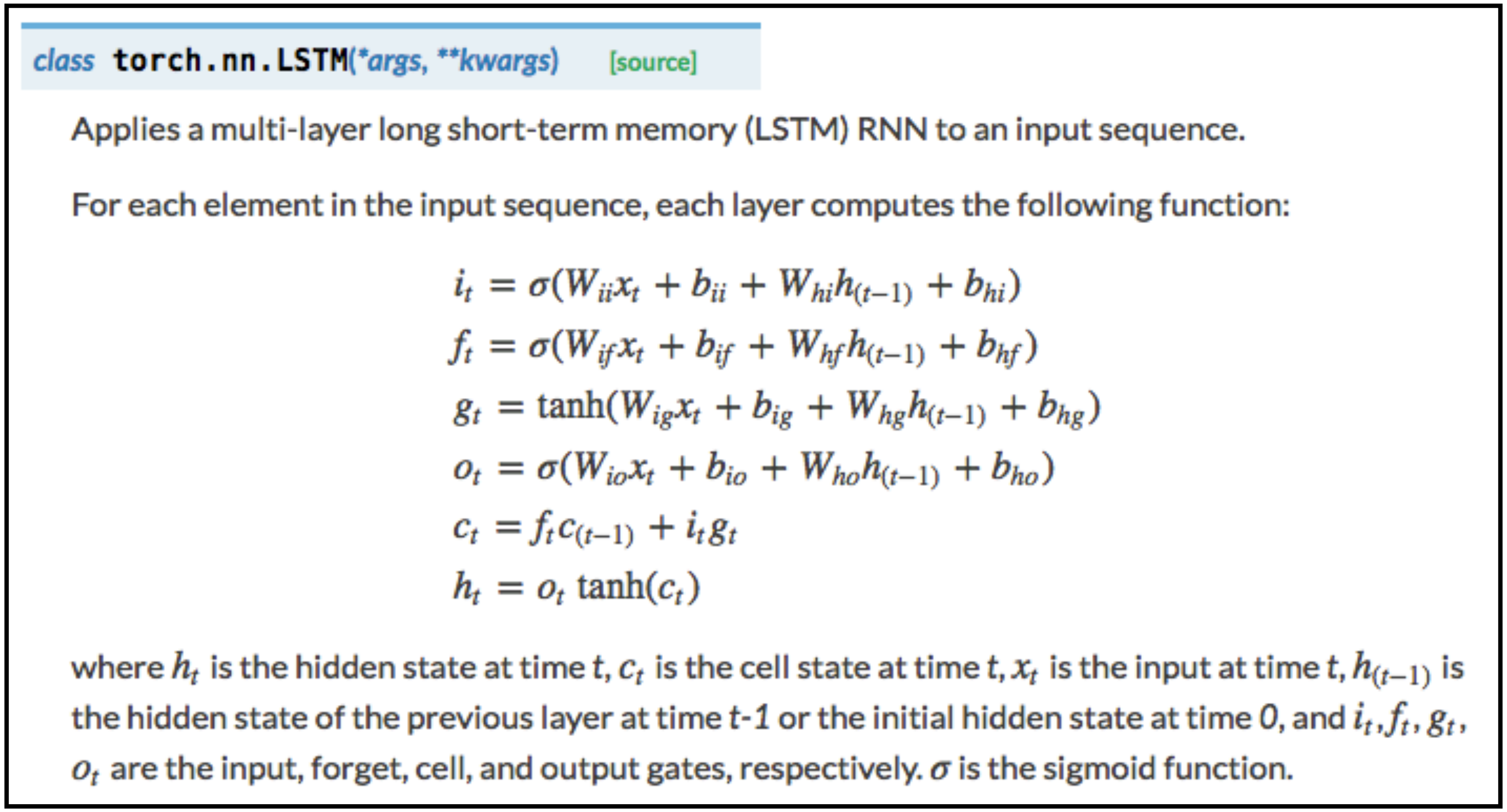
Sentiment Analysis

Negative

PyTorch RNN

- RNN, LSTM, GRU, etc.
  https://pytorch.org/docs/stable/nn.html#recurrent-layers

- Some official examples: [link]



class **torch.nn.LSTM**(**args, **kwargs*) [source]

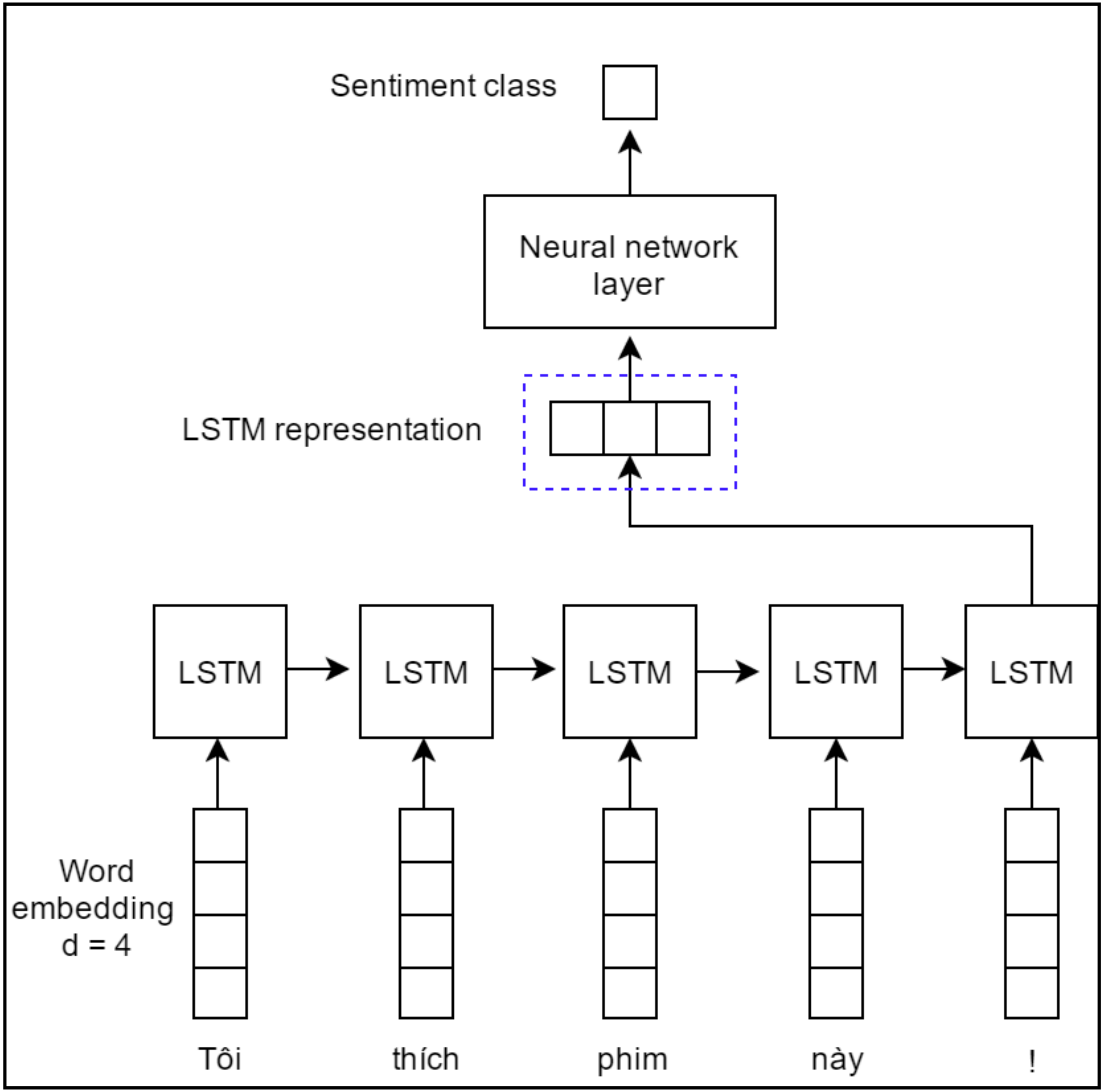
Applies a multi-layer long short-term memory (LSTM) RNN to an input sequence.

For each element in the input sequence, each layer computes the following function:

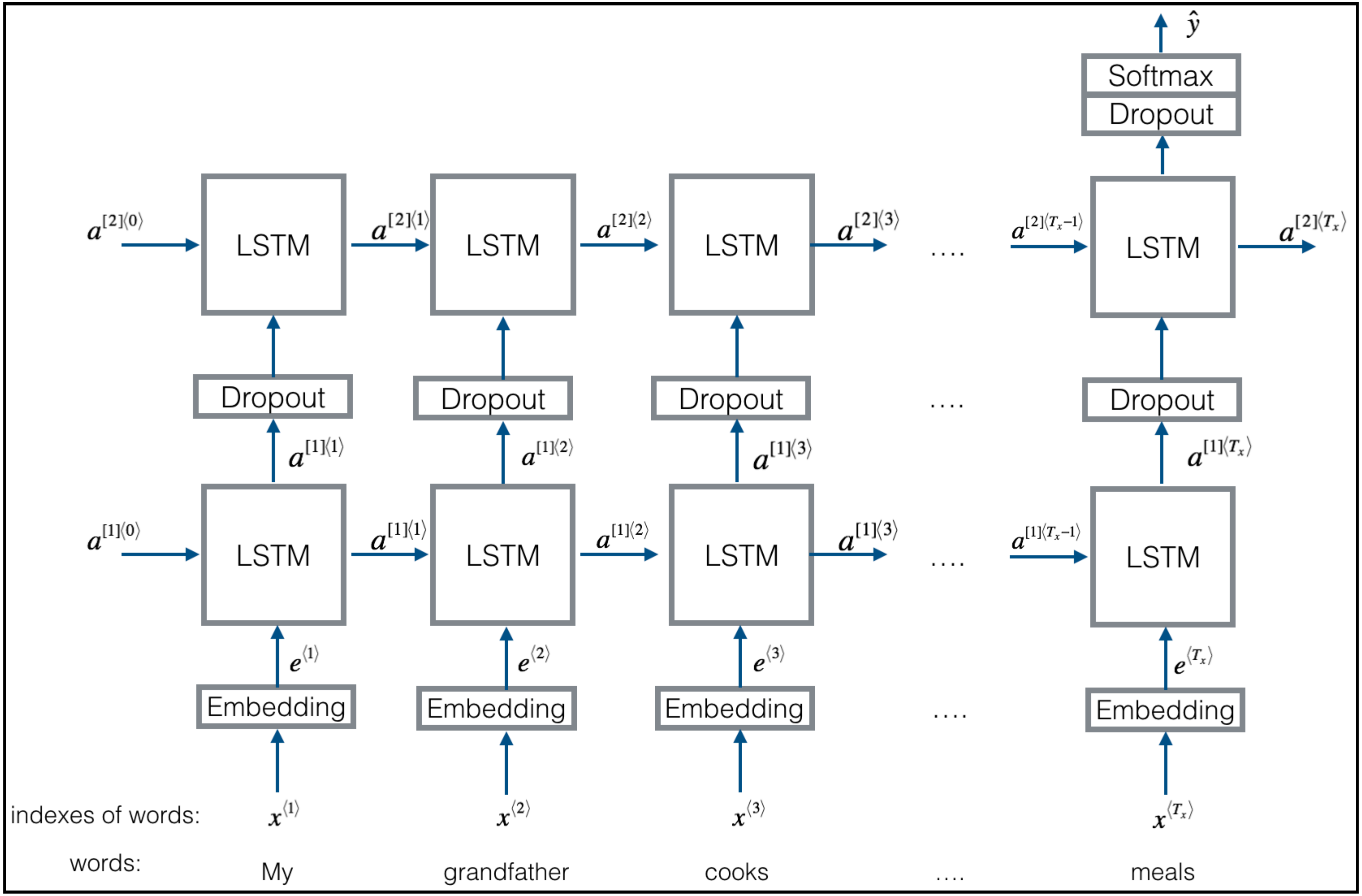
$$
\begin{aligned}
i_t &= \sigma(W_{ii} x_t + b_{ii} + W_{hi} h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if} x_t + b_{if} + W_{hf} h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig} x_t + b_{ig} + W_{hg} h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io} x_t + b_{io} + W_{ho} h_{(t-1)} + b_{ho}) \\
c_t &= f_t c_{(t-1)} + i_t g_t \\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

where h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the input at time t, $h_{(t-1)}$ is the hidden state of the previous layer at time t-1 or the initial hidden state at time 0, and i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, respectively. σ is the sigmoid function.

Example


Sentiment class
Neural network layer
LSTM representation
LSTM
LSTM
LSTM
LSTM
LSTM
Word embedding d = 4
Tôi
thích
phim
này
!

Example



$\hat{y}$

Softmax

Dropout

$a^{[2]\langle 0\rangle}$ LSTM $a^{[2]\langle 1\rangle}$ LSTM $a^{[2]\langle 2\rangle}$ LSTM $a^{[2]\langle 3\rangle}$ …. $a^{[2]\langle T_x-1\rangle}$ LSTM $a^{[2]\langle T_x\rangle}$

Dropout Dropout Dropout …. Dropout

$a^{[1]\langle 1\rangle}$ $a^{[1]\langle 2\rangle}$ $a^{[1]\langle 3\rangle}$ $a^{[1]\langle T_x\rangle}$

$a^{[1]\langle 0\rangle}$ LSTM $a^{[1]\langle 1\rangle}$ LSTM $a^{[1]\langle 2\rangle}$ LSTM $a^{[1]\langle 3\rangle}$ …. $a^{[1]\langle T_x-1\rangle}$ LSTM

$e^{\langle 1\rangle}$ $e^{\langle 2\rangle}$ $e^{\langle 3\rangle}$ $e^{\langle T_x\rangle}$

Embedding Embedding Embedding …. 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

Homework-5

- **Sentence-level Sentiment Classification with PyTorch**

- No implementation limits. It all depends on you !
(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.

- Explain your network and record the results in your **report**
(results must including the final test accuracy)

- Plagiarism (from the internet) is not permitted.