Лабораторная работа №7. Рекуррентные нейронные сети для анализа текста

Данные: Набор данных для предсказания оценок для отзывов, собранных с сайта imdb.com, который состоит из 50,000 отзывов в виде текстовых файлов. Отзывы разделены на положительные (25,000) и отрицательные (25,000). Данные предварительно токенизированы по принципу “мешка слов”, индексы слов можно взять из словаря (imdb.vocab). Обучающая выборка включает в себя 12,500 положительных и 12,500 отрицательных отзывов, контрольная выборка также содержит 12,500 положительных и 12,500 отрицательных отзывов, а также. Данные можно скачать по ссылке https://ai.stanford.edu/~amaas/data/sentiment/ (https://ai.stanford.edu/~amaas/data/sentiment/)

In [1]:

```
import torch
from torchtext import data
from torchtext import datasets
import torch.optim as optim
import torch.nn as nn
from torch.utils.tensorboard import SummaryWriter
```

Загрузите данные. Преобразуйте текстовые файлы во внутренние структуры данных, которые используют индексы вместо слов.

In [3]:

```
TEXT = data.Field(tokenize = 'spacy', include_lengths = True)
LABEL = data.LabelField(dtype = torch.float)
```

In [4]:

```
train_data, test_data = datasets.IMDB.splits(TEXT, LABEL)
print(f'Number of training examples: {len(train_data)}')
print(f'Number of testing examples: {len(test_data)}')
```

```
downloading aclImdb_v1.tar.gz

aclImdb_v1.tar.gz: 100%|██████████| 84.1M/84.1M [00:02<00:00, 33.5
MB/s]

Number of training examples: 25000
Number of testing examples: 25000
```

In [5]:

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

```
{'text': ['Clint', 'Eastwood', 'would', 'star', 'again', 'as', 'th
e', 'battle', '-', 'weary', 'Detective', 'Harry', 'Callahan', ',',
'but', 'would', 'also', 'direct', 'the', 'fourth', 'entry', 'in',
'the', "'", 'Dirty', 'Harry', "'", 'series', '.', "'", 'Sudden', '
```

Impact', "'", 'again', 'like', 'the', 'other', 'additions', ',', '
brings', 'its', 'own', 'distinguishable', 'style', 'and', 'tone',
',', 'but', 'if', 'anything', 'it', "'s", 'probably', 'the', 'most
', 'similar', 'to', 'the', 'original', 'in', 'it', "'s", 'darker',
'and', 'seedy', 'moments', '(', 'and', 'bestowing', 'a', 'classic'
, 'line', '"', 'Go', 'ahead', '.', 'Make', 'my', 'day', '"', ')',
'\x85 ', 'but', 'some', 'of', 'its', 'humor', 'has', 'to', 'been',
'seen', 'to', 'believe', '.', 'A', 'bulldog', '\x85 ', 'named', 'm
eathead', 'that', 'pisses', 'and', 'farts', '.', 'Oh', 'yeah', '.'
, 'However', 'an', 'interesting', 'fact', 'this', 'entry', 'was',
'only', 'one', 'in', 'series', 'to', 'not', 'have', 'it', 'set', '
entirely', 'in', 'San', 'Francisco.<br', '/><br', '/>The', 'story'
, 'follows', 'that', 'of', 'detective', 'Callahan', 'trying', 'to'
, 'put', 'the', 'pieces', 'together', 'of', 'a', 'murder', 'where'
, 'the', 'victim', 'was', 'shot', 'in', 'the', 'groin', 'and', 'th
en', 'between', 'the', 'eyes', '.', 'After', 'getting', 'in', 'som
e', 'trouble', 'with', 'office', 'superiors', 'and', 'causing', 'a
', 'stir', 'which', 'has', 'some', 'crime', 'lord', 'thugs', 'afte
r', 'his', 'blood', '.', 'He', "'s", 'ordered', 'to', 'take', 'lea
ve', ',', 'but', 'it', 'falls', 'into', 'a', 'working', 'one', 'wh
ere', 'he', 'heads', 'to', 'a', 'coastal', 'town', 'San', 'Paulo',
',', 'where', 'a', 'murder', 'has', 'occurred', 'similar', 'in', '
vein', '(', 'bullet', 'to', 'groin', 'and', 'between', 'eyes', ')'
, 'to', 'his', 'case', '.', 'There', 'he', 'begins', 'to', 'dig',
'up', 'dirt', ',', 'which', 'leads', 'to', 'the', 'idea', 'of', 's
omeone', 'looking', 'for', 'revenge.<br', '/><br', '/>To', 'be', '
honest', ',', 'I', 'was', "n't", 'all', 'that', 'crash', 'hot', 'o
n', 'Eastwood', "'s", 'take', ',', 'but', 'after', 'many', 'repeat
', 'viewings', 'it', 'virtually', 'has', 'grown', 'on', 'me', 'to'
, 'the', 'point', 'of', 'probably', 'being', 'on', 'par', 'with',
'the', 'first', 'sequel', "'", 'Magnum', 'Force', "'", '.', 'This'
, 'well', '-', 'assembled', 'plot', 'actually', 'gives', 'Eastwood
', 'another', 'angle', 'to', 'work', 'upon', '(', 'even', 'though'
, 'it', 'feels', 'more', 'like', 'a', 'sophisticated', 'take', 'on
', 'the', 'vigilante', 'features', 'running', 'rampant', 'at', 'th
at', 'time', ')', ',', 'quite', 'literal', 'with', 'something', 'p
unishing', 'but', 'luridly', 'damaging', '.', 'It', "'s", 'like',
'he', "'s", 'experimenting', 'with', 'noir', '-', 'thriller', 'tou
ches', 'with', 'character', '-', 'driven', 'traits', 'to', 'help',
'develop', 'the', 'emotionally', 'bubbling', 'and', 'eventual', 'm
orality', 'framework', '.', 'His', 'use', 'of', 'images', 'is', 'l
asting', ',', 'due', 'to', 'its', 'slickly', 'foreboding', 'atmosp
herics', '.', 'Dark', 'tones', ',', 'brooding', 'lighting', '\x85
', 'like', 'the', 'scene', 'towards', 'the', 'end', 'akin', 'to',
'some', 'western', 'showdown', 'of', 'a', 'silhouette', 'figure',
'(', 'Harry', 'with', 'his', 'new', '.44', 'automag', 'handgun', '
)', 'moving', 'its', 'way', 'towards', 'the', 'stunned', 'prey', '
on', 'the', 'fishing', 'docks', '.', 'It', "'s", 'a', 'striking',
'sight', 'that', 'builds', 'fear', '!', 'Mixing', 'the', 'haunting
ly', 'cold', 'with', 'plain', 'brutality', 'and', 'dash', 'of', 'h
umor', '.', 'It', 'seemed', 'to', 'come', 'off', '.', 'A', 'major'
, 'plus', 'with', 'these', 'films', 'are', 'the', 'dialogues', ','
, 'while', 'I', 'would', "n't", 'call', "'", 'Sudden', 'Impact', "
'", 'first', '-', 'rate', ',', 'it', 'provides', 'ample', 'biting'
, 'exchanges', 'and', 'memorably', 'creditable', 'lines', '\x85 ',
'"', 'You', "'re", 'a', 'legend', 'in', 'your', 'own', 'mind', '"'
, '.', 'Do', "n't", 'you', 'just', 'love', 'hearing', 'Harry', 'sp

```
arking', 'an', 'amusing', 'quip', ',', 'before', 'pulling', 'out',
'his', 'piece', '.', 'The', 'beating', 'action', 'when', 'it', 'oc
curs', 'is', 'excitingly', 'jarring', 'and', 'intense', '\x85 ', '
the', 'only', 'way', 'to', 'go', 'and', 'the', 'pacing', 'flies',
'by', 'with', 'little', 'in', 'the', 'way', 'of', 'flat', 'passage
s', '.', 'Lalo', 'Schfrin', 'would', 'return', 'as', 'composer', '
(', 'after', "'", 'The', 'Enforcer', "'", 'had', 'Jerry', 'Fieldin
g', 'scoring', ')', 'bringing', 'a', 'methodical', 'funky', 'kick'
, ',', 'which', 'still', 'breathed', 'those', 'gloomy', 'cues', 't
o', 'a', 'texturally', 'breezy', 'score', 'that', 'clicked', 'from
', 'the', 'get', '-', 'go', '.', 'Bruce', 'Surtees', '(', 'an', 'E
astwood', 'regular', ')', 'gets', 'the', 'job', 'behind', 'the', '
camera', '(', 'where', 'he', 'did', 'a', 'piecing', 'job', 'with',
"'", 'Dirty', 'Harry', "'", ')', 'and', 'gives', 'the', 'film', 'p
lenty', 'of', 'scope', 'by', 'wonderfully', 'framing', 'the', 'bac
kdrops', 'in', 'some', 'impeccable', 'tracking', 'scenes', ',', 'b
ut', 'also', 'instrument', 'edgy', 'angles', 'within', 'those', 'd
ramatic', 'moments.<br', '/><br', '/>Eastwood', 'as', 'the', 'dino
saur', 'Callahan', 'still', 'packs', 'a', 'punch', ',', 'going', '
beyond', 'just', 'that', 'steely', 'glare', 'to', 'get', 'the', 'j
ob', 'done', 'and', 'probably', 'showing', 'a', 'little', 'more',
'heart', 'than', 'one', 'would', 'expect', 'from', 'a', 'younger',
'Callahan', '.', 'This', 'going', 'by', 'the', 'sudden', 'shift',
'in', 'a', 'plot', 'turn', 'of', 'Harry', "'s", 'quest', 'for', 'j
ustice', '\x85 ', 'by', 'the', 'badge', 'even', 'though', 'he', 'd
oes', "n't", 'always', 'agree', 'with', 'it', '.', 'I', 'just', 'f
ound', 'it', 'odd', '\x85 ', 'a', 'real', 'change', 'of', 'heart',
'.', 'Across', 'from', 'him', 'is', 'a', 'stupendous', 'performanc
e', 'by', 'his', 'beau', 'at', 'the', 'time', 'Sondra', 'Locke', '
.', 'Her', 'turn', 'of', 'traumatic', 'torment', '(', 'being', 'se
nselessly', 'raped', 'along', 'with', 'her', 'younger', 'sister',
')', ',', 'is', 'hidden', 'by', 'a', 'glassily', 'quiet', 'intensi
ty', '.', 'When', 'the', 'anger', 'is', 'released', ',', 'it', "'s
", 'tactically', 'accurate', 'in', 'its', 'outcome', '.', 'Paul',
'Drake', 'is', 'perfectly', 'menacing', 'and', 'filthy', 'as', 'on
e', 'of', 'the', 'targeted', 'thugs', 'and', 'Audrie', 'J.', 'Neen
an', 'nails', 'down', 'a', 'repellently', 'scummy', 'and', 'big',
'-', 'mouthed', 'performance', '.', 'These', 'people', 'are', 'tru
ly', 'an', 'ugly', 'bunch', 'of', 'saps', '.', 'Pat', 'Hingle', 'i
s', 'sturdy', 'as', 'the', 'Chief', 'of', 'the', 'small', 'coastal
', 'town', '.', 'In', 'smaller', 'parts', 'are', 'Bradford', 'Dill
man', 'and', 'the', 'agreeably', 'potent', 'Albert', 'Popwell', '(
', 'a', 'regular', 'in', 'the', 'series', '1', '-', '4', ',', 'but
', 'under', 'different', 'characters', ')', '.', 'How', 'can', 'yo
u', 'forget', 'him', 'in', "'", 'Dirty', 'Harry', "'", '\x85 ', 'y
es', 'he', 'is', 'bank', 'robber', 'that', "'s", 'at', 'the', 'end
', 'of', 'the', 'trademark', 'quote', '"', 'Do', 'I', 'feel', 'luc
ky', '?', 'Well', ',', 'do', 'ya', ',', 'punk', '?', '"'], 'label'
: 'pos'}
```

In [6]:

```
train_data, valid_data = train_data.split()
```

In [7]:

```
print(f'Number of training examples: {len(train_data)}')
print(f'Number of validation examples: {len(valid_data)}')
```

```
Number of training examples: 17500
Number of validation examples: 7500
```

In [11]:

```
vocab_size = 25000

TEXT.build_vocab(train_data, max_size = vocab_size)
LABEL.build_vocab(train_data)
```

In [12]:

```
TEXT.vocab.freqs.most_common(20)
```

Out[12]:

```
[('the', 204019),
 (',', 193654),
 ('.', 166331),
 ('and', 110329),
 ('a', 110150),
 ('of', 101409),
 ('to', 94242),
 ('is', 76799),
 ('in', 61400),
 ('I', 54426),
 ('it', 54053),
 ('that', 49499),
 ('"', 44109),
 ("'s", 43703),
 ('this', 42479),
 ('-', 37154),
 ('/><br', 36011),
 ('was', 35256),
 ('as', 30408),
 ('with', 30268)]
```

In [13]:

```
TEXT.vocab.itos[:10]
```

Out[13]:

```
['<unk>', '<pad>', 'the', ',', '.', 'and', 'a', 'of', 'to', 'is']
```

Реализуйте и обучите двунаправленную рекуррентную сеть (LSTM или GRU).

In [14]:

```
batch_size = 32
device = 'cuda'

train_iterator, valid_iterator, test_iterator = data.BucketIterator.splits(
    (train_data, valid_data, test_data),
    batch_size = batch_size,
    device = device
)
```

In [15]:

```
input_size = len(TEXT.vocab)
embedding_size = 100
hidden_size = 256
output_size = 1
learning_rate = 0.001
pad_idx = TEXT.vocab.stoi[TEXT.pad_token]
```

In [50]:

```
from model import RNN

model = RNN(input_size, embedding_size, hidden_size, output_size, pad_idx)
```

In [17]:

```
optimizer = torch.optim.AdamW(model.parameters(), lr=learning_rate)
logs_writer = SummaryWriter(log_dir='./logs/rnn')
criterion = nn.BCEWithLogitsLoss()
```

In [18]:

```
model = model.to(device)
criterion = criterion.to(device)
```

In [25]:

```
from train import train

train(model, train_iterator, valid_iterator, optimizer, criterion, logs_writer
, 5)
```

```
100%|██████████| 5/5 [11:03<00:00, 132.71s/it]
```

In [27]:

```
test_loss, test_acc = evaluate(model, test_iterator, criterion)

print(f'Test Loss: {test_loss:.3f} | Test Acc: {test_acc*100:.2f}%')
```

```
Test Loss: 0.482 | Test Acc: 77.93%
```

Используйте индексы слов и их различное внутреннее представление (word2vec, glove).

In [51]:

```
from torchtext.vocab import GloVe
# build the vocabulary
embedding_size = 100
TEXT.build_vocab(train_data, max_size = vocab_size, vectors=GloVe(name='6B', d
im=embedding_size))
LABEL.build_vocab(train_data)
```

In [52]:

```
model = RNN(input_size, embedding_size, hidden_size, output_size, pad_idx)
```

In [53]:

```
pretrained_embeddings = TEXT.vocab.vectors

print(pretrained_embeddings.shape)
```

```
torch.Size([25002, 100])
```

In [54]:

```
model.embedding.weight.data.copy_(pretrained_embeddings)
```

Out[54]:

```
tensor([[ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.000
0],
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.000
0],
        [-0.0382, -0.2449,  0.7281,  ..., -0.1459,  0.8278,  0.270
6],
        ...,
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.000
0],
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.000
0],
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.000
0]])
```

In [55]:

```
pad_idx = TEXT.vocab.stoi[TEXT.pad_token]
unk_idx = TEXT.vocab.stoi[TEXT.unk_token]

model.embedding.weight.data[unk_idx] = torch.zeros(embedding_size)
model.embedding.weight.data[pad_idx] = torch.zeros(embedding_size)

print(model.embedding.weight.data)
```

```
tensor([[ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.000
0],
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.000
0],
        [-0.0382, -0.2449,  0.7281,  ..., -0.1459,  0.8278,  0.270
6],
        ...,
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.000
0],
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.000
0],
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.000
0]])
```

In [56]:

```
batch_size = 32
device = 'cuda'

train_iterator, valid_iterator, test_iterator = data.BucketIterator.splits(
    (train_data, valid_data, test_data),
    batch_size = batch_size,
    device = device
)
```

In [57]:

```
input_size = len(TEXT.vocab)
hidden_size = 256
output_size = 1
learning_rate = 0.001
```

In [58]:

```
optimizer = torch.optim.AdamW(model.parameters(), lr=learning_rate)
logs_writer = SummaryWriter(log_dir='./logs/glove')
criterion = nn.BCEWithLogitsLoss()
```

In [59]:

```
model = model.to(device)
criterion = criterion.to(device)
```

In [60]:

```
train(model, train_iterator, valid_iterator, optimizer, criterion, logs_writer
, 5)
```

```
  0%|          | 0/5 [00:00<?, ?it/s]
 20%|██        | 1/5 [02:12<08:48, 132.11s/it]
 40%|████      | 2/5 [04:25<06:37, 132.45s/it]
 60%|██████    | 3/5 [06:37<04:24, 132.34s/it]
 80%|████████  | 4/5 [08:51<02:12, 132.91s/it]
100%|██████████| 5/5 [11:06<00:00, 133.23s/it]
```

In [61]:

```
test_loss, test_acc = evaluate(model, test_iterator, criterion)

print(f'Test Loss: {test_loss:.3f} | Test Acc: {test_acc*100:.2f}%')
```

```
Test Loss: 0.356 | Test Acc: 87.19%
```

Поэкспериментируйте со структурой сети (добавьте больше рекуррентных, полносвязных или сверточных слоев).

In [71]:

```
from model import RNN2

model = RNN2(input_size, embedding_size, hidden_size, output_size, pad_idx)
```

In [72]:

```
pretrained_embeddings = TEXT.vocab.vectors
model.embedding.weight.data.copy_(pretrained_embeddings)

pad_idx = TEXT.vocab.stoi[TEXT.pad_token]
unk_idx = TEXT.vocab.stoi[TEXT.unk_token]

model.embedding.weight.data[unk_idx] = torch.zeros(embedding_size)
model.embedding.weight.data[pad_idx] = torch.zeros(embedding_size)
```

In [75]:

```
optimizer = torch.optim.AdamW(model.parameters(), lr=learning_rate)
logs_writer = SummaryWriter(log_dir='./logs/rnn-additional_fc')
criterion = nn.BCEWithLogitsLoss()
```

In [76]:

```
model = model.to(device)
criterion = criterion.to(device)
```

In [77]:

```
train(model, train_iterator, valid_iterator, optimizer, criterion, logs_writer
, 5)
```

```
  0%|          | 0/5 [00:00<?, ?it/s]

 20%|██        | 1/5 [02:13<08:54, 133.55s/it]

 40%|████      | 2/5 [04:27<06:40, 133.63s/it]

 60%|██████    | 3/5 [06:40<04:26, 133.44s/it]

 80%|████████  | 4/5 [08:54<02:13, 133.75s/it]

100%|██████████| 5/5 [11:08<00:00, 133.78s/it]
```

In [78]:

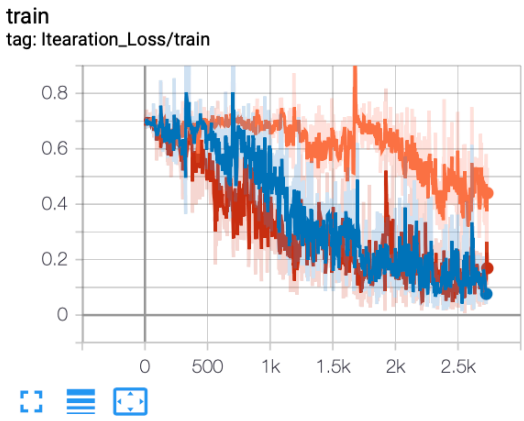
```
test_loss, test_acc = evaluate(model, test_iterator, criterion)

print(f'Test Loss: {test_loss:.3f} | Test Acc: {test_acc*100:.2f}%')
```

```
Test Loss: 0.407 | Test Acc: 84.31%
```

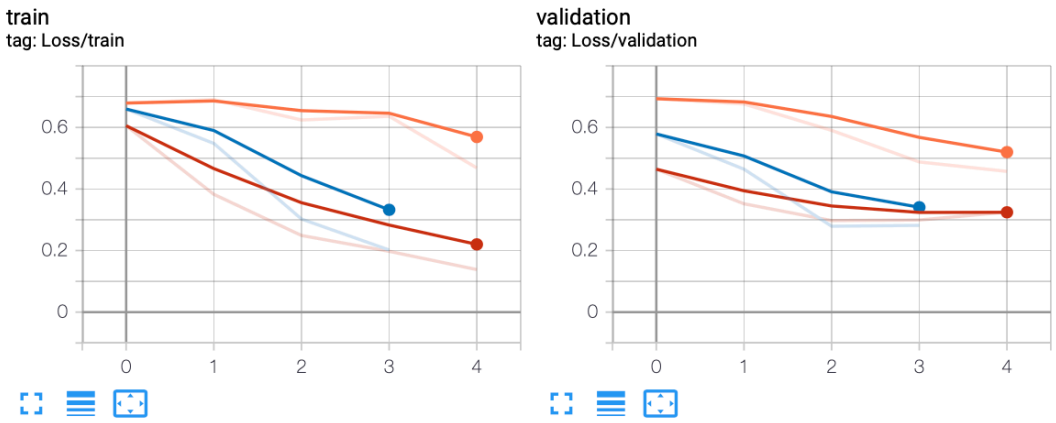
Accuracy

Itearation_Loss



Name	Smoothed	Value	Step	Time	Relative
glove	0.858	0.8919	3	Sat Apr 18, 18:59:21	6m 39s
rnn	0.7335	0.7888	4	Sat Apr 18, 18:21:41	8m 52s
rnn-additional-fc	0.8758	0.8742	4	Sat Apr 18, 19:39:02	8m 55s

Loss



Вывод:

В данной работе была реализованная рекуррентная сеть для анализа текста, были использованные разные модели сети и способы векторизации текста.