**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 [0]:
```
!pip install git+https://github.com/d2l-ai/d2l-en # installing
!pip install -U --pre mxnet-cu101mkl  # updating mxnet to at le

```

```
    Uninstalling pyzmq-17.0.0:
      Successfully uninstalled pyzmq-17.0.0
Successfully installed d2l-0.11.4 pyzmq-19.0.0

Collecting mxnet-cu101mkl
  Downloading https://files.pythonhosted.org/packages/3d/4b/e51dc49ca5fe6564028e7c91b10a3f79c00d710dd691b408c77597df5883/mxnet_cu101mkl-1.6.0-py2.py3-none-manylinux1_x86_64.whl (https://files.pythonhosted.org/packages/3d/4b/e51dc49ca5fe6564028e7c91b10a3f79c00d710dd691b408c77597df5883/mxnet_cu101mkl-1.6.0-py2.py3-none-manylinux1_x86_64.whl) (711.0MB)
     |████████████████████████████████| 711.0MB 24kB/s
Collecting graphviz<0.9.0,>=0.8.1
  Downloading https://files.pythonhosted.org/packages/53/39/4ab213673844e0c004bed8a0781a0721a3f6bb23eb8854ee75c236428892/graphviz-0.8.4-py2.py3-none-any.whl (https://files.pythonhosted.org/packages/53/39/4ab213673844e0c004bed8a0781a0721a3f6bb23eb8854ee75c236428892/graphviz-0.8.4-py2.py3-none-any.whl)
Requirement already satisfied, skipping upgrade: numpy<2.0.0,>1.16
```

In [0]:
```
import shutil
from google.colab import drive
import d2l
from mxnet import gluon, np, npx, init
import os
from mxnet.gluon import nn, rnn
from mxnet.contrib import text
npx.set_np()

```

**Задание 1.** Загрузите данные. Преобразуйте текстовые файлы во внутренние структуры данных, которые используют индексы вместо слов.

In [0]:
```
!wget https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.
drive.mount('/content/drive')
shutil.unpack_archive("aclImdb_v1.tar.gz", "/content/aclImdb_v1
```

```
--2020-04-03 17:29:45--  https://ai.stanford.edu/~amaas/data/senti
ment/aclImdb_v1.tar.gz
(https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz)
Resolving ai.stanford.edu (ai.stanford.edu)... 171.64.68.10
Connecting to ai.stanford.edu (ai.stanford.edu)|171.64.68.10|:443.
.. connected.
HTTP request sent, awaiting response... 200 OK
Length: 84125825 (80M) [application/x-gzip]
Saving to: ‘aclImdb_v1.tar.gz.2’

aclImdb_v1.tar.gz.2 100%[===================>]  80.23M  85.2MB/s
in 0.9s

2020-04-03 17:29:46 (85.2 MB/s) - ‘aclImdb_v1.tar.gz.2’ saved [841
25825/84125825]

Drive already mounted at /content/drive; to attempt to forcibly re
mount, call drive.mount("/content/drive", force_remount=True).
```

In [0]:
```
def read_imdb(data_dir, is_train):
    data, labels = [], []
    for label in ('pos', 'neg'):
        folder_name = os.path.join(data_dir, 'train' if is_trai
                                   label)
        for file in os.listdir(folder_name):
            with open(os.path.join(folder_name, file), 'rb') as
                review = f.read().decode('utf-8').replace('\n',
                data.append(review)
                labels.append(1 if label == 'pos' else 0)
    return data, labels

train_data = read_imdb('/content/aclImdb_v1/aclImdb', is_train=
print('trainings:', len(train_data[0]))
```

```
trainings: 25000
```

In [0]:
```
train_tokens = d2l.tokenize(train_data[0], token='word')
vocab = d2l.Vocab(train_tokens, min_freq=5, reserved_tokens=['<

d2l.set_figsize((3.5, 2.5))
d2l.plt.hist([len(line) for line in train_tokens], bins=range(0
```

```
<Figure size 252x180 with 1 Axes>
```

In [0]:
```
num_steps = 500  # sequence length
train_features = np.array([d2l.truncate_pad(
    vocab[line], num_steps, vocab['<pad>']) for line in train_t
train_features.shape
```

Out[13]:
```
(25000, 500)
```

In [0]:
```
train_iter = d2l.load_array((train_features, train_data[1]), 64
'batches:', len(train_iter)
```

Out[17]:
```
('batches:', 391)
```

In [0]:
```
def load_data_imdb(batch_size, num_steps=500):
    data_dir = '/content/aclImdb_v1/aclImdb'
    train_data = read_imdb(data_dir, True)
    test_data = read_imdb(data_dir, False)
    train_tokens = d2l.tokenize(train_data[0], token='word')
    test_tokens = d2l.tokenize(test_data[0], token='word')
    vocab = d2l.Vocab(train_tokens, min_freq=5)
    train_features = np.array([d2l.truncate_pad(
        vocab[line], num_steps, vocab.unk) for line in train_to
    test_features = np.array([d2l.truncate_pad(
        vocab[line], num_steps, vocab.unk) for line in test_tok
    train_iter = d2l.load_array((train_features, train_data[1])
    test_iter = d2l.load_array((test_features, test_data[1]), b
                               is_train=False)
    return train_iter, test_iter, vocab
```

**Задание 2.** Реализуйте и обучите двунаправленную рекуррентную сеть (LSTM или GRU). Какого качества классификации удалось достичь?

In [0]:
```
batch_size = 64
train_iter, test_iter, vocab = load_data_imdb(batch_size)
```

In [0]:
```
class BiRNN(nn.Block):
    def __init__(self, vocab_size, embed_size, num_hiddens,
                 num_layers, **kwargs):
        super(BiRNN, self).__init__(**kwargs)
        self.embedding = nn.Embedding(vocab_size, embed_size)
        self.encoder = rnn.LSTM(num_hiddens, num_layers=num_lay
                                bidirectional=True, input_size=
        self.decoder = nn.Dense(2)

    def forward(self, inputs):
        embeddings = self.embedding(inputs.T)
        outputs = self.encoder(embeddings)
        encoding = np.concatenate((outputs[0], outputs[-1]), ax
        outs = self.decoder(encoding)
        return outs
```

In [0]:
```
embed_size, num_hiddens, num_layers, ctx = 100, 100, 2, d2l.try
net = BiRNN(len(vocab), embed_size, num_hiddens, num_layers)
net.initialize(init.Xavier(), ctx=ctx)
```

In [0]:
```
lr, num_epochs = 0.01, 5
trainer = gluon.Trainer(net.collect_params(), 'adam', {'learnin
loss = gluon.loss.SoftmaxCrossEntropyLoss()
d2l.train_ch13(net, train_iter, test_iter, loss, trainer, num_e
```

```
loss 0.095, train acc 0.970, test acc 0.823
580.3 examples/sec on [gpu(0)]

<Figure size 252x180 with 1 Axes>
```

In [0]:
```
predict_sentiment(net, vocab, 'this movie is so great')
```

Out[54]:
```
'positive'
```

In [0]:
```
predict_sentiment(net, vocab, 'this movie is so bad')
```

Out[55]:
```
'negative'
```

**Задание 3.** Используйте индексы слов и их различное внутреннее представление (word2vec, glove). Как влияет данное преобразование на качество классификации?

In [0]:
```
glove_embedding = text.embedding.create('glove', pretrained_fil
```

```
Downloading /root/.mxnet/embeddings/glove/glove.6B.zip from https://apache-mxnet.s3-accelerate.dualstack.amazonaws.com/gluon/embeddings/glove/glove.6B.zip... (https://apache-mxnet.s3-accelerate.dualstack.amazonaws.com/gluon/embeddings/glove/glove.6B.zip...)
```

In [0]:
```
embeds = glove_embedding.get_vecs_by_tokens(vocab.idx_to_token)
embeds.shape
```

Out[42]:
```
(49339, 100)
```

In [0]:
```
net.embedding.weight.set_data(embeds)
net.embedding.collect_params().setattr('grad_req', 'null')
```

In [0]:
```
lr, num_epochs = 0.01, 5
trainer = gluon.Trainer(net.collect_params(), 'adam', {'learnin
loss = gluon.loss.SoftmaxCrossEntropyLoss()
d2l.train_ch13(net, train_iter, test_iter, loss, trainer, num_e
```

```
loss 0.317, train acc 0.866, test acc 0.844
604.6 examples/sec on [gpu(0)]

<Figure size 252x180 with 1 Axes>
```

In [0]:
```
def predict_sentiment(net, vocab, sentence):
    sentence = np.array(vocab[sentence.split()], ctx=d2l.try_gp
    label = np.argmax(net(sentence.reshape(1, -1)), axis=1)
    return 'positive' if label == 1 else 'negative'
```

In [0]:
```
predict_sentiment(net, vocab, 'this movie is so great')
```

Out[50]:
```
'positive'
```

In [0]:
```
predict_sentiment(net, vocab, 'this movie is so bad')
```

Out[51]:
```
'negative'
```

**Задание 4.** Поэкспериментируйте со структурой сети (добавьте больше рекуррентных, полносвязных или сверточных слоев). Как это повлияло на качество классификации?

In [0]:
```
embed_size, num_hiddens, num_layers, ctx = 100, 100, 4, d2l.try
net = BiRNN(len(vocab), embed_size, num_hiddens, num_layers)
net.initialize(init.Xavier(), ctx=ctx)
```

In [0]:
```
lr, num_epochs = 0.01, 5
trainer = gluon.Trainer(net.collect_params(), 'adam', {'learnin
loss = gluon.loss.SoftmaxCrossEntropyLoss()
d2l.train_ch13(net, train_iter, test_iter, loss, trainer, num_e
```

```
loss 0.075, train acc 0.975, test acc 0.840
265.2 examples/sec on [gpu(0)]

<Figure size 252x180 with 1 Axes>
```

**Задание 5.** Используйте предобученную рекуррентную нейронную сеть (например, DeepMoji или что-то подобное). Какой максимальный результат удалось получить на контрольной выборке?

In [2]:
```
!git clone https://github.com/bfelbo/DeepMoji.git
```

```
Cloning into 'DeepMoji'...
remote: Enumerating objects: 281, done.
remote: Total 281 (delta 0), reused 0 (delta 0), pack-reused 281
Receiving objects: 100% (281/281), 110.54 MiB | 28.08 MiB/s, done.
Resolving deltas: 100% (142/142), done.
Checking out files: 100% (66/66), done.
```

In [17]:
```
!pip install tensorflow==1.14
```

```
Collecting tensorflow==1.14
  Downloading https://files.pythonhosted.org/packages/de/f0/96fb2e
0412ae9692dbf400e5b04432885f677ad6241c088ccc5fe7724d69/tensorflow-
1.14.0-cp36-cp36m-manylinux1_x86_64.whl
(https://files.pythonhosted.org/packages/de/f0/96fb2e0412ae9692dbf
400e5b04432885f677ad6241c088ccc5fe7724d69/tensorflow-1.14.0-cp36-c
p36m-manylinux1_x86_64.whl) (109.2MB)
     |████████████████████████████████| 109.2MB 97kB/s
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/
local/lib/python3.6/dist-packages (from tensorflow==1.14) (1.1.0)
Requirement already satisfied: astor>=0.6.0 in /usr/local/lib/pyth
on3.6/dist-packages (from tensorflow==1.14) (0.8.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/pytho
n3.6/dist-packages (from tensorflow==1.14) (1.12.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/
python3.6/dist-packages (from tensorflow==1.14) (1.1.0)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/pytho
n3.6/dist-packages (from tensorflow==1.14) (0.34.2)
Collecting tensorboard<1.15.0,>=1.14.0
  Downloading https://files.pythonhosted.org/packages/91/2d/2ed263
```

```
449a078cd9c8a9ba50ebd50123adf1f8cfbea1492f9084169b89d9/tensorboard
-1.14.0-py3-none-any.whl
(https://files.pythonhosted.org/packages/91/2d/2ed263449a078cd9c8a
9ba50ebd50123adf1f8cfbea1492f9084169b89d9/tensorboard-1.14.0-py3-n
one-any.whl) (3.1MB)
     |████████████████████████████████| 3.2MB 49.4MB/s
Requirement already satisfied: gast>=0.2.0 in /usr/local/lib/pytho
n3.6/dist-packages (from tensorflow==1.14) (0.3.3)
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/py
thon3.6/dist-packages (from tensorflow==1.14) (0.9.0)
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/pyt
hon3.6/dist-packages (from tensorflow==1.14) (1.12.1)
Collecting tensorflow-estimator<1.15.0rc0,>=1.14.0rc0
  Downloading https://files.pythonhosted.org/packages/3c/d5/21860a
5b11caf0678fbc8319341b0ae21a07156911132e0e71bffed0510d/tensorflow_
estimator-1.14.0-py2.py3-none-any.whl
(https://files.pythonhosted.org/packages/3c/d5/21860a5b11caf0678fb
c8319341b0ae21a07156911132e0e71bffed0510d/tensorflow_estimator-1.1
4.0-py2.py3-none-any.whl) (488kB)
     |████████████████████████████████| 491kB 56.5MB/s
Requirement already satisfied: protobuf>=3.6.1 in /usr/local/lib/p
ython3.6/dist-packages (from tensorflow==1.14) (3.10.0)
Requirement already satisfied: google-pasta>=0.1.6 in /usr/local/l
ib/python3.6/dist-packages (from tensorflow==1.14) (0.2.0)
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/pyt
hon3.6/dist-packages (from tensorflow==1.14) (1.27.2)
Requirement already satisfied: numpy<2.0,>=1.14.5 in /usr/local/li
b/python3.6/dist-packages (from tensorflow==1.14) (1.18.2)
Requirement already satisfied: keras-applications>=1.0.6 in /usr/l
ocal/lib/python3.6/dist-packages (from tensorflow==1.14) (1.0.8)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/p
ython3.6/dist-packages (from tensorboard<1.15.0,>=1.14.0->tensorfl
ow==1.14) (3.2.1)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib
/python3.6/dist-packages (from tensorboard<1.15.0,>=1.14.0->tensor
flow==1.14) (1.0.1)
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/li
b/python3.6/dist-packages (from tensorboard<1.15.0,>=1.14.0->tenso
rflow==1.14) (46.0.0)
Requirement already satisfied: h5py in /usr/local/lib/python3.6/di
st-packages (from keras-applications>=1.0.6->tensorflow==1.14) (2.
10.0)
Installing collected packages: tensorboard, tensorflow-estimator,
tensorflow
  Found existing installation: tensorboard 2.2.0
    Uninstalling tensorboard-2.2.0:
      Successfully uninstalled tensorboard-2.2.0
  Found existing installation: tensorflow-estimator 2.2.0rc0
    Uninstalling tensorflow-estimator-2.2.0rc0:
      Successfully uninstalled tensorflow-estimator-2.2.0rc0
  Found existing installation: tensorflow 2.2.0rc2
    Uninstalling tensorflow-2.2.0rc2:
      Successfully uninstalled tensorflow-2.2.0rc2
```

```
Successfully installed tensorboard-1.14.0 tensorflow-1.14.0 tensor
flow-estimator-1.14.0
```

In [0]:

```
import os
os.chdir('/content/DeepMoji')
```

In [3]:

```
from __future__ import print_function
import deepmoji
import examples.example_helper
import numpy as np
from keras.preprocessing import sequence
from keras.datasets import imdb
import deepmoji.model_def

nb_tokens = 20000
maxlen = 80
batch_size = 32

print('Loading data...')
print(len(train_data[0]), 'train sequences')
print(len(test_data[0]), 'test sequences')
print('Pad sequences (samples x time)')
print('X_train shape:', train_data.shape)
print('X_test shape:', test_data.shape)

print('Build model...')
model = deepmoji.model_def.deepmoji_architecture(nb_classes=2,
model.summary()

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

print('Train...')
model.fit(train_data[0], train_data[1], batch_size=batch_size,
          validation_data=(test_data[0], test_data[0]))
score, acc = model.evaluate(test_data[0], test_data[1], batch_s
print('Test score:', score)
print('Test accuracy:', acc)
```

```
Loading data...
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
X_train shape: (25000, 80)
X_test shape: (25000, 80)
```

```
Build model...
Model: "DeepMoji"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
Connected to
=================================================================
input_2 (InputLayer)         (None, 80)                0
_________________________________________________________________
embedding (Embedding)        (None, 80, 256)           5120000
input_2[0][0]
_________________________________________________________________
activation_2 (Activation)    (None, 80, 256)           0
embedding[0][0]
_________________________________________________________________
bi_lstm_0 (Bidirectional)    (None, 80, 1024)          3149824
activation_2[0][0]
_________________________________________________________________
bi_lstm_1 (Bidirectional)    (None, 80, 1024)          6295552
bi_lstm_0[0][0]
_________________________________________________________________
concatenate_2 (Concatenate)  (None, 80, 2304)          0
bi_lstm_1[0][0]
                                                       b
i_lstm_0[0][0]
                                                       a
ctivation_2[0][0]
_________________________________________________________________
attlayer (AttentionWeightedAver (None, 2304)           2304
concatenate_2[0][0]
_________________________________________________________________
softmax (Dense)              (None, 1)                 2305
attlayer[0][0]
=================================================================
Total params: 14,569,985
Trainable params: 14,569,985
Non-trainable params: 0
_________________________________________________________________
Train...
Train on 25000 samples, validate on 25000 samples
Epoch 1/5
25000/25000 [==============================] - 6155s 246ms/step -
loss: 0.4263 - acc: 0.8038 - val_loss: 0.3644 - val_acc: 0.8446
```

```
Epoch 2/5
25000/25000 [==============================] - 5949s 238ms/step -
loss: 0.2494 - acc: 0.9028 - val_loss: 0.3690 - val_acc: 0.8428
Epoch 3/5
25000/25000 [==============================] - 5676s 227ms/step -
loss: 0.1577 - acc: 0.9453 - val_loss: 0.4109 - val_acc: 0.8388
Epoch 4/5
25000/25000 [==============================] - 5633s 225ms/step -
loss: 0.0913 - acc: 0.9715 - val_loss: 0.5532 - val_acc: 0.8300
Epoch 5/5
25000/25000 [==============================] - 5886s 235ms/step -
loss: 0.0647 - acc: 0.9823 - val_loss: 0.7103 - val_acc: 0.8306
25000/25000 [==============================] - 1033s 41ms/step
Test score: 0.7102789056491852
Test accuracy: 0.83064
```