Лабораторная работа №7. Рекуррентные нейронные сети для анализа текста

1. Загрузите данные. Преобразуйте текстовые файлы во внутренние структуры данных, которые используют индексы вместо слов.
2. Реализуйте и обучите двунаправленную рекуррентную сеть (LSTM или GRU). Какого качества классификации удалось достичь?
3. Используйте индексы слов и их различное внутреннее представление (word2vec, glove). Как влияет данное преобразование на качество классификации?
4. Поэкспериментируйте со структурой сети (добавьте больше рекуррентных, полносвязных или сверточных слоев). Как это повлияло на качество классификации?
5. Используйте предобученную рекуррентную нейронную сеть (например, DeepMoji или что-то подобное).

```
try:
  import wget
except:
  !pip install wget
  import wget
```

```
Collecting wget
  Downloading https://files.pythonhosted.org/packages/47/6a/62e288da7bcda82b935ff0c6cfe542970f04e29c756b0e147251b2fb2
Building wheels for collected packages: wget
  Building wheel for wget (setup.py) ... done
  Created wheel for wget: filename=wget-3.2-cp36-none-any.whl size=9682 sha256=4cd6f942b03df0d209ec08406371b48342c253
  Stored in directory: /root/.cache/pip/wheels/40/15/30/7d8f7cea2902b4db79e3fea550d7d7b85ecb27ef992b618f3f
Successfully built wget
Installing collected packages: wget
Successfully installed wget-3.2
```

```
!wget https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
!wget  http://nlp.stanford.edu/data/glove.6B.zip
```

```
--2020-04-19 12:45:15--  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’

aclImdb_v1.tar.gz   100%[===================>]  80.23M  20.3MB/s    in 7.2s

2020-04-19 12:45:23 (11.1 MB/s) - ‘aclImdb_v1.tar.gz’ saved [84125825/84125825]
```

```
!ls data/aclImdb/train
```

```
labeledBow.feat  pos    unsupBow.feat  urls_pos.txt
neg              unsup  urls_neg.txt   urls_unsup.txt
```

```
import tarfile
with tarfile.open('aclImdb_v1.tar.gz') as tar:
  tar.extractall('data')
```

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

```
import pandas as pd
import glob
import os
import string

def get_dfs(start_path):

  df = pd.DataFrame(columns=['text', 'sent'])
  text = []
  sent = []
  for p in ['pos','neg']:
```

```
    for p in ['pos', 'neg']:
      path=os.path.join(start_path, p)
      files = [f for f in os.listdir(path)
               if os.path.isfile(os.path.join(path,f))]
      for f in files:
        with open (os.path.join(path, f), "r") as myfile:
          # replace carriage return linefeed with spaces
          text.append(myfile.read()
                      .replace("\n", " ")
                      .replace("\r", " "))
          # convert positive reviews to 1 and negative reviews to zero
          sent.append(1 if p == 'pos' else 0)

    df['text']=text
    df['sent']=sent
    #This line shuffles the data so you don't end up with contiguous
    #blocks of positive and negative reviews
    df = df.sample(frac=1).reset_index(drop=True)
    return df

train_df = get_dfs ("data/aclImdb/train/")
test_df = get_dfs ("data/aclImdb/test/")

train_df.head()
```

	text	sent
0	Don't really know where to start with one of t...	0
1	Watching Smother was perhaps the longest not-q...	0
2	I hadn't heard of this film until I read an ar...	0
3	Perhaps one of the most overrated so-called ho...	0
4	I like my Ronald Colman dashing and debonair, ...	1

```
import tensorflow as tf
NUM_WORDS=20000
```

```
SEQ_LEN=100
EMBEDDING_SIZE=100
BATCH_SIZE=128
EPOCHS=5
THRESHOLD=0.5

#create tokenizer for our data
tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=NUM_WORDS, oov_token='<UNK>')
tokenizer.fit_on_texts(train_df['text'])


#convert text data to numerical indexes
train_seqs=tokenizer.texts_to_sequences(train_df['text'])
test_seqs=tokenizer.texts_to_sequences(test_df['text'])

#pad data up to SEQ_LEN (note that we truncate if there are more than SEQ_LEN tokens)
train_seqs=tf.keras.preprocessing.sequence.pad_sequences(train_seqs, maxlen=SEQ_LEN, padding="post")
test_seqs=tf.keras.preprocessing.sequence.pad_sequences(test_seqs, maxlen=SEQ_LEN, padding="post")
```

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

79 процентов

```
model = tf.keras.Sequential([
  tf.keras.layers.Embedding(NUM_WORDS, EMBEDDING_SIZE),
  tf.keras.layers.GlobalAveragePooling1D(),
  tf.keras.layers.Dense(1, activation='sigmoid')])

model.summary()

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

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_17 (Embedding)     (None, None, 100)         2000000
_________________________________________________________________
global_average_pooling1d_2 ( (None, 100)               0
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 101
=================================================================
Total params: 2,000,101
Trainable params: 2,000,101
Non-trainable params: 0
_________________________________________________________________
```

```
es = tf.keras.callbacks.EarlyStopping(monitor='accuracy', mode='max')
callbacks=[es]
history = model.fit(train_seqs, train_df['sent'].values,
                    batch_size=BATCH_SIZE, epochs=EPOCHS, validation_split=0.2, callbacks=callbacks)
```

```
Epoch 1/5
157/157 [==============================] - 4s 24ms/step - loss: 0.6551 - accuracy: 0.7160 - val_loss: 0.5918 - val_acc
Epoch 2/5
157/157 [==============================] - 4s 23ms/step - loss: 0.5192 - accuracy: 0.8109 - val_loss: 0.4643 - val_acc
Epoch 3/5
157/157 [==============================] - 4s 23ms/step - loss: 0.4076 - accuracy: 0.8515 - val_loss: 0.3969 - val_acc
Epoch 4/5
157/157 [==============================] - 4s 23ms/step - loss: 0.3402 - accuracy: 0.8749 - val_loss: 0.3609 - val_acc
Epoch 5/5
157/157 [==============================] - 4s 23ms/step - loss: 0.2960 - accuracy: 0.8913 - val_loss: 0.3415 - val_acc
```

```
model.evaluate(test_seqs, test_df['sent'].values)
```

```
782/782 [==============================] - 2s 2ms/step - loss: 0.3542 - accuracy: 0.8465
[0.3541618883609772, 0.8464800119400024]
```

```
model = tf.keras.Sequential([
  tf.keras.layers.Embedding(NUM_WORDS, EMBEDDING_SIZE),
```

```
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(6)),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(1, activation='sigmoid')])

model.summary()

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

history = model.fit(train_seqs, train_df['sent'].values,
                    batch_size=BATCH_SIZE, epochs=EPOCHS, validation_split=0.2, callbacks=callbacks)
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_28 (Embedding)     (None, None, 100)         2000000
_________________________________________________________________
bidirectional_14 (Bidirectio (None, 12)                5136
_________________________________________________________________
dropout_5 (Dropout)          (None, 12)                0
_________________________________________________________________
dense_29 (Dense)             (None, 1)                 13
=================================================================
Total params: 2,005,149
Trainable params: 2,005,149
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
157/157 [==============================] - 6s 37ms/step - loss: 0.5797 - accuracy: 0.7204 - val_loss: 0.4087 - val_ac
Epoch 2/5
157/157 [==============================] - 5s 33ms/step - loss: 0.3317 - accuracy: 0.8735 - val_loss: 0.3695 - val_ac
Epoch 3/5
157/157 [==============================] - 5s 33ms/step - loss: 0.2371 - accuracy: 0.9179 - val_loss: 0.4023 - val_ac
```

```
model.evaluate(test_seqs, test_df['sent'].values)
```

```
782/782 [==============================] - 5s 6ms/step - loss: 0.4252 - accuracy: 0.7986
```

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

Качество модели повысилось до 82 процентов

```
!wget  http://nlp.stanford.edu/data/glove.6B.zip
```

```
--2020-04-19 14:01:01--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
--2020-04-19 14:01:01--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2020-04-19 14:01:02--  http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip’

glove.6B.zip        100%[===================>] 822.24M  2.16MB/s    in 6m 29s

2020-04-19 14:07:31 (2.11 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]
```

```
!unzip glove.6B.zip
!ls
```

```
Archive:  glove.6B.zip
  inflating: glove.6B.50d.txt
  inflating: glove.6B.100d.txt
  inflating: glove.6B.200d.txt
  inflating: glove.6B.300d.txt
aclImdb_v1.tar.gz  glove.6B.100d.txt  glove.6B.300d.txt  glove.6B.zip
data               glove.6B.200d.txt  glove.6B.50d.txt   sample_data
```

```
import numpy as np
embeddings_index = {}
with open('glove.6B.100d.txt') as f:
    for line in f:
        word, coefs = line.split(maxsplit=1)
        coefs = np.fromstring(coefs, 'f', sep=' ')
        embeddings_index[word] = coefs

print('Found %s word vectors.' % len(embeddings_index))
```

```
Found 400000 word vectors.
```

```
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))

num_words = min(NUM_WORDS, len(word_index) + 1)
embedding_matrix = np.zeros((num_words, 100))
for word, i in word_index.items():
    #print (word)
    if i >= NUM_WORDS:
        continue
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # words not found in embedding index will be all-zeros.
        embedding_matrix[i] = embedding_vector
```

```
Found 88583 unique tokens.
```

```
model = tf.keras.Sequential([
  tf.keras.layers.Embedding(NUM_WORDS, EMBEDDING_SIZE),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(60)),
  tf.keras.layers.Dropout(0.25),
  tf.keras.layers.Dense(1, activation='sigmoid')])

model.layers[0].set_weights([embedding_matrix])
model.layers[0].trainable = False
```

```
model.summary()

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

history = model.fit(train_seqs, train_df['sent'].values,
                    batch_size=BATCH_SIZE, epochs=EPOCHS, validation_split=0.2, callbacks=callbacks)
```

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_30 (Embedding)     (None, None, 100)         2000000
_________________________________________________________________
bidirectional_16 (Bidirectio (None, 120)               77280
_________________________________________________________________
dropout_7 (Dropout)          (None, 120)               0
_________________________________________________________________
dense_31 (Dense)             (None, 1)                 121
=================================================================
Total params: 2,077,401
Trainable params: 77,401
Non-trainable params: 2,000,000
_________________________________________________________________
Epoch 1/5
157/157 [==============================] - 3s 19ms/step - loss: 0.5761 - accuracy: 0.6877 - val_loss: 0.4987 - val_ac
Epoch 2/5
157/157 [==============================] - 2s 15ms/step - loss: 0.4688 - accuracy: 0.7785 - val_loss: 0.4474 - val_ac
Epoch 3/5
157/157 [==============================] - 2s 15ms/step - loss: 0.4399 - accuracy: 0.7943 - val_loss: 0.4179 - val_ac
Epoch 4/5
157/157 [==============================] - 2s 15ms/step - loss: 0.4073 - accuracy: 0.8120 - val_loss: 0.4046 - val_ac
Epoch 5/5
157/157 [==============================] - 2s 14ms/step - loss: 0.3926 - accuracy: 0.8214 - val_loss: 0.3955 - val_ac
```

```
model.evaluate(test_seqs, test_df['sent'].values)
```

```
782/782 [==============================] - 5s 6ms/step - loss: 0.3995 - accuracy: 0.8204
[0.3994566798210144, 0.820360004901886]
```