# 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/

**Задание 1.**

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

In [132]:

```
from tensorflow import keras
import numpy as np
from keras.datasets import imdb

epochs = 3
vocab_size = 10000
review_max_len = 200
feature_count = 50
batch_size = 256

data_folder = '../data'
```

In [133]:

```
(X_train, y_train), (X_dev, y_dev) = imdb.load_data(num_words=vocab_size)
```

In [134]:

```
X_train = keras.preprocessing.sequence.pad_sequences(X_train, maxlen=review_max_len)
X_dev = keras.preprocessing.sequence.pad_sequences(X_dev, maxlen=review_max_len)
```

**Задание 2.**

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

In [135]:

```
def model_factory():
    return keras.models.Sequential([
        keras.layers.Embedding(vocab_size, feature_count),
        keras.layers.Bidirectional(keras.layers.GRU(units=32)),
        keras.layers.Dense(1, activation='sigmoid')
    ])
```

In [136]:

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

    model.fit(X_train, y_train, validation_data=(X_dev, y_dev), epochs=epochs,
              batch_size=batch_size)
```

In [137]:

```
model = model_factory()
train(model)
```

```
train(model)
```

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/3
25000/25000 [==============================] - 44s 2ms/sample - loss: 0.5740 - accuracy: 0.6835 -
val_loss: 0.3851 - val_accuracy: 0.8334
Epoch 2/3
25000/25000 [==============================] - 41s 2ms/sample - loss: 0.3020 - accuracy: 0.8782 -
val_loss: 0.3530 - val_accuracy: 0.8475
Epoch 3/3
25000/25000 [==============================] - 42s 2ms/sample - loss: 0.2151 - accuracy: 0.9199 -
val_loss: 0.3296 - val_accuracy: 0.8626
```

**Задание 3.**

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

In [138]:

```
embeddings_index = dict()
with open(data_folder + '/glove6b/glove.6B.50d.txt') as f:
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs
```

In [139]:

```
embedding_matrix = np.zeros((vocab_size, feature_count))
for word, i in filter(lambda elem : elem[1] < vocab_size, imdb.get_word_index().items()):
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

In [140]:

```
def model_factory():
    return keras.models.Sequential([
        keras.layers.Embedding(vocab_size, feature_count, weights=[embedding_matrix],
                               input_length=review_max_len, trainable=False),
        keras.layers.Bidirectional(keras.layers.GRU(units=32)),
        keras.layers.Dense(1, activation='sigmoid')
    ])
```

In [141]:

```
model = model_factory()
train(model)
```

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/3
25000/25000 [==============================] - 45s 2ms/sample - loss: 0.6927 - accuracy: 0.5249 -
val_loss: 0.6879 - val_accuracy: 0.5415
Epoch 2/3
25000/25000 [==============================] - 65s 3ms/sample - loss: 0.6814 - accuracy: 0.5608 -
val_loss: 0.6783 - val_accuracy: 0.5674
Epoch 3/3
25000/25000 [==============================] - 47s 2ms/sample - loss: 0.6640 - accuracy: 0.5962 -
val_loss: 0.6570 - val_accuracy: 0.6102
```

**Задание 4.**

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

In [142]:

```
def model_factory():
    return keras.models.Sequential([
        keras.layers.Embedding(vocab_size, feature_count, weights=[embedding_matrix],
                               input_length=review_max_len),
        keras.layers.Bidirectional(keras.layers.GRU(units=32)),
        keras.layers.Dense(64, activation='relu'),
        keras.layers.Dense(1, activation='sigmoid')
    ])
```

In [143]:

```
model = model_factory()
train(model)
```

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/3
25000/25000 [==============================] - 56s 2ms/sample - loss: 0.6908 - accuracy: 0.5320 -
val_loss: 0.6780 - val_accuracy: 0.5682
Epoch 2/3
25000/25000 [==============================] - 52s 2ms/sample - loss: 0.5856 - accuracy: 0.6790 -
val_loss: 0.4389 - val_accuracy: 0.7983
Epoch 3/3
25000/25000 [==============================] - 46s 2ms/sample - loss: 0.3468 - accuracy: 0.8489 -
val_loss: 0.3425 - val_accuracy: 0.8511
```