Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных сетей и систем

Кафедра Информатики

ЛАБОРАТОРНАЯ РАБОТА №7
« Рекуррентные нейронные сети для анализа текста»

Магистрант:
гр. 956241
Шуба И.А.

Проверил:
Заливако С. С.

Минск, 2020

ХОД РАБОТЫ

Задание.

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.

Задание 2.

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

Задание 3.

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

Задание 4.

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

Задание 5.

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

Результат выполнения:

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

Укажем путь и загрузим датасет.

```
path = 'aclImdb/train/'
```

```
sentence_list = []
labels = []

print(f"Reading Positive dataset: {path}+pos/*.txt")
for filename in os.listdir(path+'pos/'):
    filepath = f"{path}/pos/{filename}"
    with open(filepath, "r") as fp:
        txt = fp.read().replace("<br />", "\n")
        sentence_list.append(txt)
        labels.append(1)

print(f"Reading Positive dataset: {path}+neg/*.txt")
for filename in os.listdir(path+'neg/'):
    filepath = f"{path}/neg/{filename}"
    with open(filepath, "r") as fp:
        txt = fp.read().replace("<br />", "\n")
        sentence_list.append(txt)
        labels.append(0)
```

```
dataset = pd.DataFrame(data={'review': sentence_list, 'label': labels})
```

```
dataset.shape
```

Out[5]:

```
(25000, 2)
```

Рисунок 1 – Загрузка данных

Пример одного отзыва.

```
dataset['review'][3]
```

Out[6]:

```
'This is an early film "Pilot" for the hit Canadian tv show Trailer Park B
oys. It was played to executives at a few networks before Showcase decided
to sign them up for a tv series. Great acting and a very funny cast make t
his one of the best cult comedy films. The movie plot is that these two sm
all time criminals go around "exterminating" peoples pets for money. If yo
u have a dog next door whos barking all night these are the guys you go t
o! But they get into trouble when they come across a job too big for them
to deal with and end up in a shootout. Watch this movie if you want to und
erstand the beginning of the tv series. I highly recommend it!\n\nRated R
for swearing, violence, and drug use.\n\nIts not too offensive either (the
y dont actually show killing animals)'
```

Рисунок 2 – Пример одного отзыва

Напишем функцию для обработки текста.

```
TAG_RE = re.compile(r'<[^>]+>')
def remove_tags(text):
    return TAG_RE.sub('', text)

def preprocessing_text(text):
    # Remove html tag
    sentence = remove_tags(text)
    # Remove link
    sentence = re.sub(r'https:\/\/[a-zA-Z]*\.com',' ',sentence)
    # Remove number
    sentence = re.sub(r'\d+',' ',sentence)
    # Remove white space
    sentence = re.sub(r'\s+',' ',sentence)
    # Remove single character
    sentence = re.sub(r"\b[a-zA-Z]\b", ' ', sentence)
    # Remove bracket
    sentence = re.sub(r'\W+',' ',sentence)
    # Make sentence lowercase
    sentence = sentence.lower()
    return sentence
```

Рисунок 3 – Функция для обработки текста

Выполним препроцессинг и посмотрим на результат обработки.

```
pre_proces_sen = []
sentences = list(dataset['review'])
for sen in tqdm(sentences):
    pre_proces_sen.append(preprocessing_text(sen))
```

```
100%|██████████| 25000/25000 [00:05<00:00, 4813.03it/s]
```

```
pre_proces_sen[5]
```

```
Out[9]:

' cannot accept the negative comments of other reviewers they are too crit
ical perhaps because they are stuck in the past would like to see comment
from someone who had never seen basic instinct perhaps someone very young
left the cinema feeling glad that had not been swayed by the imdb reviewer
s hours later am still trying to find flaws in the plot but cannot think o
f anything serious my advice to everyone is see it for yourself and make u
p your own mind it follows similar pattern to basic instinct but the plot
is less confused it still left me wondering at the end but in more satisfa
ctory way sharon stone is as sexy and evil as before and wears her years e
xtremely well this remains her defining role david morrisey was satisfacto
ry even though he is no michael douglas of the supporting cast particularl
y liked david thewlis as the police detective '
```

Рисунок 4 – Препроцессинг текста и отображение примера после обработки

Импортируем необходимые библиотеки для удаления «stopwords».

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

```
[nltk_data] Downloading package stopwords to /home/aksel/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /home/aksel/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
stopwords=set(stopwords.words('english'))
```

Рисунок 5 – Импорт необходимых библиотек для обработки текста

Из отзывов удалим «stopwords».

```
for i in tqdm(range(len(pre_proces_sen))):
    x = pre_proces_sen[i]
    x = word_tokenize(x)
    new_x_list = [word for word in x if word not in stopwords]
    pre_proces_sen[i] = ' '.join(new_x_list)
```

```
100%|██████████| 25000/25000 [00:17<00:00, 1397.15it/s]
```

Рисунок 6 – Удаление «stopwords»

Пример отзыва после обработки.

```
pre_proces_sen[5]
```

```
Out[12]:

'accept negative comments reviewers critical perhaps stuck past would like
see comment someone never seen basic instinct perhaps someone young left c
inema feeling glad swayed imdb reviewers hours later still trying find fla
ws plot think anything serious advice everyone see make mind follows simil
ar pattern basic instinct plot less confused still left wondering end sati
sfactory way sharon stone sexy evil wears years extremely well remains def
ining role david morrisey satisfactory even though michael douglas support
ing cast particularly liked david thewlis police detective'
```

Рисунок 6 –Примера отзыва после обработки

```
dataset['review'] = pre_proces_sen
```

```
dataset.head()
```

Out[15]:

	review	label
0	william shakespeare would proud particular ver...	1
1	half dozen short stories varying interest enli...	1
2	easy call guys dolls great got frank sinatra m...	1
3	early film pilot hit canadian tv show trailer ...	1
4	delightful disney film angela lansbury fine fo...	1

Рисунок 7 – Замена исходного датасета на обработанный

Разделение подготовленного датасета на контрольную, валидационную и тренировочную выборки.

```
X_train, X_test, y_train, y_test = train_test_split(dataset['review'].values,
                                                    dataset['label'].values, test_size=
0.20, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, ran
dom_state=42)
```

Рисунок 8 – Разделение датасета

Токенизация и создание последовательностей.

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(dataset['review'].values)

X_train = tokenizer.texts_to_sequences(X_train)
X_val = tokenizer.texts_to_sequences(X_val)
X_test = tokenizer.texts_to_sequences(X_test)
```

```
vocab_size = len(tokenizer.word_index) + 1
print(vocab_size)
maxlen = 100
X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
X_val = pad_sequences(X_val, padding='post', maxlen=maxlen)
X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)
```

```
73395
```

Рисунок 9 – Токенизация и создание последовательностей

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

Реализуем двунаправленную рекуррентную сеть с помощью слоя Biderectional.

```
model = Sequential()
model.add(Embedding(vocab_size, EMBED_DIM, input_length=maxlen, embeddings_regularizer=
regularizers.l2(0.001)))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(128, recurrent_dropout=0.4)))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))


model.compile(loss='binary_crossentropy',
              optimizer='adagrad',
              metrics=['acc'])

keras2ascii(model)
```

Рисунок 10 – Реализация двунаправленной рекуррентной сети

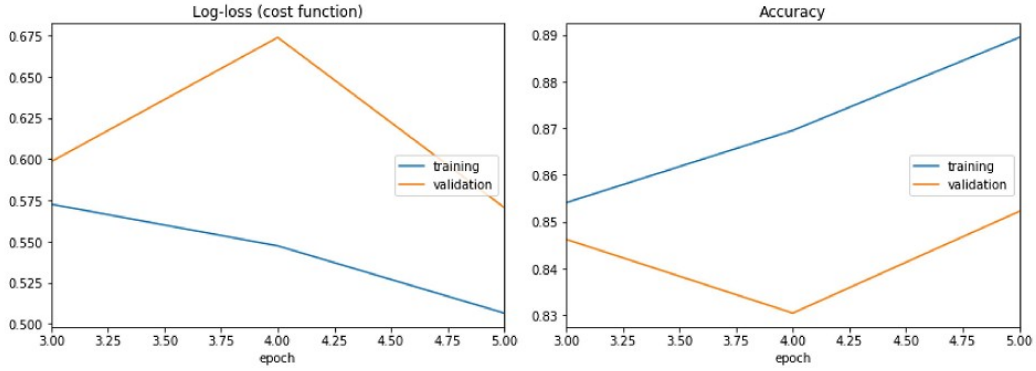
```
    OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

        Input   #####         100
    Embedding   emb | -------------------  36697500    98.3%
                #####    100  500
      Dropout    | || -------------------         0     0.0%
                #####    100  500
Bidirectional   ????? -------------------    644096     1.7%
                #####         256
      Dropout    | || -------------------         0     0.0%
                #####         256
        Dense   XXXXX -------------------       257     0.0%
      sigmoid   #####           1
```

Рисунок 11 – Схема двунаправленной рекуррентной сети

Обучение будет осуществляться с небольшим числом эпох, из-за специфики рекуррентых нейронных сетей.

```
history = tqdm(model.fit(X_train, y_train,
                         epochs=5,
                         batch_size=128,
                         validation_data=(X_val, y_val),
                         callbacks=[PlotLossesKeras()],
                         verbose=1))
```




```
Log-loss (cost function):
training   (min:    0.507, max:    1.389, cur:    0.507)
validation (min:    0.571, max:    0.674, cur:    0.571)

Accuracy:
training   (min:    0.718, max:    0.889, cur:    0.889)
validation (min:    0.804, max:    0.852, cur:    0.852)
```

Рисунок 12 – Результаты обучения

Применив всего лишь 5 эпох, были получены достаточно неоднозначные графики. Максимальная точность на валидационных данных составила 0,852, а минимальный log-loss – 0,571

.

```
y_pred = model.predict_classes(X_test)
evaluation1 = model.evaluate(X_test, y_test)
acc1 = accuracy_score(y_test, y_pred)
f1_1 = f1_score(y_test, y_pred, average='weighted')
print(evaluation1)
print('Accuracy: ', acc1)
print('F1 score: ', f1_1)
```

```
5000/5000 [==============================] - 16s 3ms/step
[0.5618007720947266, 0.851]
Accuracy:  0.851
F1 score:  0.8509417241582818
```

Рисунок 13 – Результаты модели на контрольной выборке

Таким образом, точность модели на контрольной выборке составила 0,851, а log-loss 0,561.

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

Используем word2vec в качестве другого внутреннего представления отзывов.

```
reviews = []
for i in range(len(pre_proces_sen)):
    reviews.append(pre_proces_sen[i].split())
```

```
%%time
import gensim
word2vec = gensim.models.Word2Vec(sentences=reviews,size=EMBED_DIM,
                                  window=5,workers=4,min_count=3)
```

```
CPU times: user 1min 2s, sys: 153 ms, total: 1min 2s
Wall time: 1min 2s
```

```
words=list(word2vec.wv.vocab)
print("vocabulary size:",len(words))
```

```
vocabulary size: 37248
```

```
words[:10]
```

```
Out[50]:

['william',
 'shakespeare',
 'would',
 'proud',
 'particular',
 'version',
 'play',
 'best',
 'movie',
 'also']
```

Рисунок 14 – Использование word2vec

Заполним веса, полученные с word2vec.

```
embedding_weights = np.zeros((vocab_size, EMBED_DIM))
for word, index in tokenizer.word_index.items():
    try:
        embedding_weights[index] = word2vec.wv.get_vector(word)
    except:
        pass
```

Рисунок 15 – Заполнение весов

В той же самой сети дополним поле weights в слое Embedding полученными весами с word2vec.

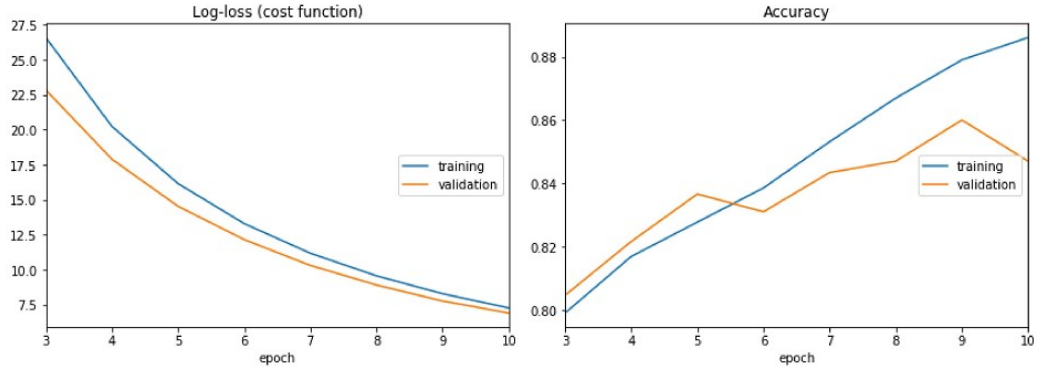
```
model = Sequential()
model.add(Embedding(vocab_size, EMBED_DIM, input_length=maxlen, embeddings_regularizer=
regularizers.l2(0.001),
                    weights=[embedding_weights]))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(128, recurrent_dropout=0.4)))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))


model.compile(loss='binary_crossentropy',
              optimizer='adagrad',
              metrics=['acc'])

keras2ascii(model)
```

Рисунок 16 – Реализация двунаправленной рекуррентной сети с добавлением полученных весов

```
history = tqdm(model.fit(X_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_data=(X_val, y_val),
                    callbacks=[PlotLossesKeras()],
                    verbose=1))
```




```
Log-loss (cost function):
training   (min:    7.267, max:   74.862, cur:    7.267)
validation (min:    6.910, max:   47.398, cur:    6.910)

Accuracy:
training   (min:    0.688, max:    0.886, cur:    0.886)
validation (min:    0.765, max:    0.860, cur:    0.847)
```

Рисунок 17 – Результаты обучения

Как видно из графиков, максимальная точность на валидационных данных составила 0,86, а минимальный log-loss – 6,910. При данном представлении слов, достаточно хорошо себя ведет log-loss. Точность модели на валидационной и тренировочной выборке ведет себя немного неоднозначно.Уже после 5 эпохи начинается заметное расхождение с

точностью на тренировочных данных, однако расхождение не такое значительное, какое было в предыдущем задании.

```
y_pred = model.predict_classes(X_test)
evaluation1 = model.evaluate(X_test, y_test)
acc1 = accuracy_score(y_test, y_pred)
f1_1 = f1_score(y_test, y_pred, average='weighted')
print(evaluation1)
print('Accuracy: ', acc1)
print('F1 score: ', f1_1)
```

```
5000/5000 [==============================] - 16s 3ms/step
[6.8980816513061525, 0.8488]
Accuracy:  0.8488
F1 score:  0.8479178939382327
```

Рисунок 18 – Результаты модели на контрольной выборке

Таким образом, точность модели на контрольной выборке составила 0,848, а log-loss 6,89, хоть эти результаты и хуже, чем на предыдущей модели, однако линии обучения ведут себя более стабильней, что является более важным при оценке адекватности модели.

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

Добавим сверточный слой, и слой пулинга.

```
model = Sequential()
model.add(Embedding(vocab_size, EMBED_DIM, input_length=maxlen,
                    weights=[embedding_weights]))
model.add(Conv1D(filters=128, kernel_size=3, padding='same', activation='relu'))
model.add(Dropout(0.5))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.4))
model.add(LSTM(64, recurrent_dropout=0.4))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='adagrad',
              metrics=['acc'])

keras2ascii(model)
```

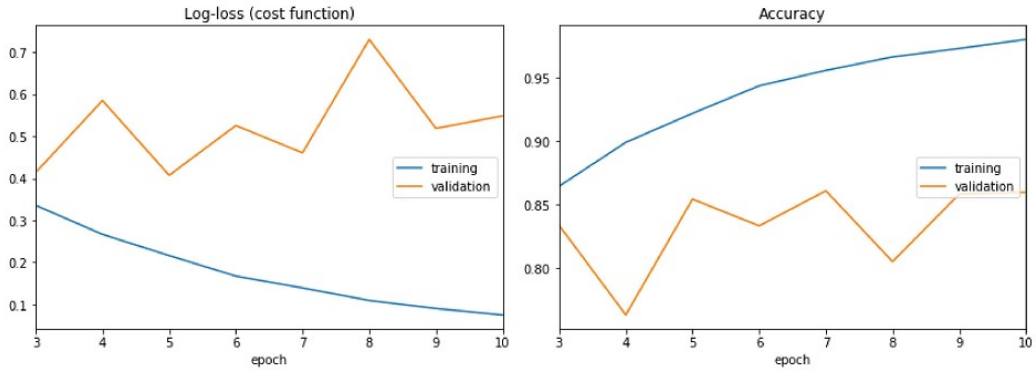
Рисунок 19 – Реализация рекуррентной нейронной сети с добавлением сверточного слоя и слоя пуллинга

```
    OPERATION          DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

        Input   #####         100
    Embedding   emb |  ------------------  36697500     99.3%
                #####    100  500
       Conv1D    \|/   ------------------    192128      0.5%
         relu   #####    100  128
      Dropout    | ||  ------------------         0      0.0%
                #####    100  128
MaxPooling1D    Y max  ------------------         0      0.0%
                #####     50  128
      Dropout    | ||  ------------------         0      0.0%
                #####     50  128
         LSTM   LLLLL  ------------------     49408      0.1%
         tanh   #####          64
      Dropout    | ||  ------------------         0      0.0%
                #####          64
        Dense   XXXXX  ------------------        65      0.0%
      sigmoid   #####           1
```

Рисунок 20 – Схема рекуррентной нейронной сети с добавлением сверточного слоя и слоя пуллинга

```
history = tqdm(model.fit(X_train, y_train,
                         epochs=10,
                         batch_size=128,
                         validation_data=(X_val, y_val),
                         callbacks=[PlotLossesKeras()],
                         verbose=1))
```




```
Log-loss (cost function):
training   (min:    0.075, max:    0.619, cur:    0.075)
validation (min:    0.407, max:    0.731, cur:    0.549)

Accuracy:
training   (min:    0.649, max:    0.980, cur:    0.980)
validation (min:    0.763, max:    0.861, cur:    0.860)
```

Рисунок 21 – Обучение рекуррентной нейронной сети

```
y_pred = model.predict_classes(X_test)
evaluation1 = model.evaluate(X_test, y_test)
acc1 = accuracy_score(y_test, y_pred)
f1_1 = f1_score(y_test, y_pred, average='weighted')
print(evaluation1)
print('Accuracy: ', acc1)
print('F1 score: ', f1_1)
```

```
5000/5000 [==============================] - 5s 992us/step
[0.5437909519642592, 0.8612]
Accuracy:  0.8612
F1 score:  0.8612005774083696
```

Рисунок 22 – Результаты модели на контрольной выборке

Таким образом, точность модели на контрольной выборке составила 0,86, а log-loss 0,54, хоть эти результаты и лучше, чем на предыдущей модели, однако линии обучения ведут себя нестабильно.

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

Используем предобученную рекуррентную нейронную сеть –DeepMoji.

```
model = deepmoji_architecture(nb_classes, vocab_size, maxlen, embed_dropout_rate=0.5, f
inal_dropout_rate=0.2,
                              embed_l2=0.001)
```

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

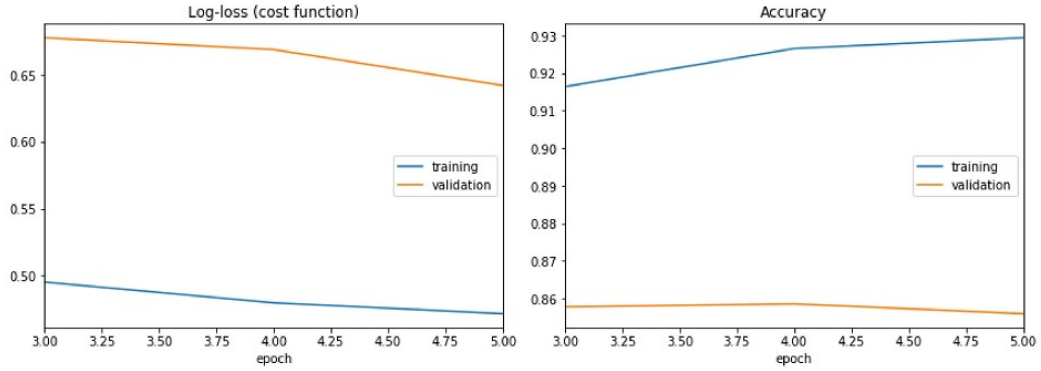
Рисунок 23 – Импорт и реализация предобученной модели DeepMoji

```
          OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

              Input   #####         100
          InputLayer     |   -------------------         0     0.0%
                      #####         100
           Embedding  emb  | -------------------  18789120    66.5%
                tanh  #####    100  256
      SpatialDropout1D  ????? -------------------         0     0.0%
                      #####    100  256
         Bidirectional  ????? -------------------   3149824    11.2%
                      #####    100 1024
         Bidirectional  ????? -------------------   6295552    22.3%
                      #####    100 1024
           Concatenate  ????? -------------------         0     0.0%
                      #####    100 2304
AttentionWeightedAverage   ????? -------------------      2304     0.0%
                      #####        2304
               Dropout    | || -------------------         0     0.0%
                      #####        2304
                 Dense   XXXXX -------------------      2305     0.0%
               sigmoid   #####           1
```

Рисунок 24 – Схема предобученной нейронной сети DeepMoji

```
history = model.fit(X_train, y_train,
                    epochs=5,
                    batch_size=128,
                    validation_data=(X_val, y_val),
                    callbacks=[PlotLossesKeras()],
                    verbose=1)
```




```
Log-loss (cost function):
training   (min:    0.471, max:    2.145, cur:    0.471)
validation (min:    0.603, max:    0.678, cur:    0.642)

Accuracy:
training   (min:    0.782, max:    0.929, cur:    0.929)
validation (min:    0.856, max:    0.868, cur:    0.856)
```

Рисунок 25 – Обучение рекуррентной нейронной сети

```
evaluation1 = model.evaluate(X_test, y_test)
print(evaluation1)
```

```
5000/5000 [==============================] - 105s 21ms/step
[0.6480377429962159, 0.85]
```

Рисунок 26 – Результаты модели на контрольной выборке

Таким образом, точность модели на контрольной выборке составила 0,85, а log-loss 0,64, хоть эти результаты и хуже, чем на предыдущей модели, однако линии обучения ведут себя более стабильней, что является более важным при оценке адекватности модели. Также отметим, что линии точности и log-loss не сходятся, а напротив идут параллельно друг другу. Можно предположить, что сети не хватает какого-то толчка выбраться из локального минимума функции, поэтому точность модели никак не изменяется с увеличением эпох.

Вывод:

В ходе выполнения лабораторной работы были изучены подходы для работы с текстом. Были реализованы различные представления текста: токенизация, создание «мешка слов», а также word2vec. Были опробованы рекуррентные архитектуры нейронных сетей, а также их комбинации с о сверточными сетями, а также со слоями, выполняющими функцию пуллинга. Так же была опробована предобученная нейронная сеть DeepMoji, которая применяется, для разделения настроения. В результате наилучшая точность модели была достигнута для контрольной выборки – 0,86 в реализации с комбинацией различных слоев.