МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Искусственные нейронные сети»

Тема: "Классификация обзоров фильмов"

Студентка гр. 8383	____________________	Аверина О.С.
Преподаватель	____________________	Жангиров Т.Р.

Санкт-Петербург

2021

Цель.

Классифицировать обзоры фильма с помощью реккурентной нейронной сети.

Постановка задачи.

1. Ознакомиться с рекуррентными нейронными сетями
2. Изучить способы классификации текста
3. Ознакомиться с ансамблированием сетей
4. Построить ансамбль сетей, который позволит получать точность не менее 97%

Требования

1. Найти набор оптимальных ИНС для классификации текста
2. Провести ансамблирование моделей
3. Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей
4. Провести тестирование сетей на своих текстах (привести в отчете)

Выполнения работы.

1. Был загружен датасет IMDb, встроенный в Keras.
```
from keras.datasets import imdb
(training_data, training_targets), (testing_data,
testing_targets) = imdb.load_data(num_words=10000)
data = np.concatenate((training_data, testing_data),
axis=0)
targets = np.concatenate((training_targets,
testing_targets), axis=0)
```

2. Датасет был разделен на обучающий и тестировочный наборы.

Обучающий набор будет состоять из 40 000 обзоров, тестировочный — из 10 000.

```
X_train = data[:train_size]
Y_train = targets[:train_size]
X_test = data[train_size:]
Y_test = targets[train_size:]
```

3. Далее были обрезаны и дополнены входные последовательности так, чтобы они были одинаковой длины для моделирования.

```
X_train = sequence.pad_sequences(X_train,
maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test,
maxlen=max_review_length)
```

4. Была создана модель 1. В качестве входного слоя был выбран слой Embedded, который использует 32 вектора длины для представления каждого слова. В качестве скрытого слоя выбран слой LSTM с 100 единицами памяти. Выходной слой - полносвязный слой с функцией активации sigmoid.

   Т.к. нужно решить проблему классификации, в качестве фунции потерь используется binary_crossentropy. Используется эффективный алгоритм оптимизации ADAM. Модель подходит только для 2 эпох, потому что она быстро решает проблему. Большой пакет из 64 обзоров используется для разметки обновлений веса.

   Далее модель была обучена и протестирована на тестовых данных.

```
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length,
input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])
print(model.summary())
model.fit(X_train, y_train, validation_data=(X_test,
y_test), epochs=3, batch_size=64)
scores = model.evaluate(X_test, y_test, verbose=0)
```

Структура модели:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 500, 32)           640000
_________________________________________________________________
lstm (LSTM)                  (None, 100)               53200
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 693,301
Trainable params: 693,301
Non-trainable params: 0
_________________________________________________________________
```

5. Модель 1 достигла точности на тестовых данных 88.77%.

```
Epoch 1/3 625/625 [==============================] - 387s 612ms/step - loss: 0.4896 - accuracy: 0.7366 - val_loss: 0.2850 - val_accuracy: 0.8852 Epoch 2/3 625/625 [==============================] - 383s 613ms/step - loss: 0.2514 - accuracy: 0.9044 - val_loss: 0.2834 - val_accuracy: 0.8839 Epoch 3/3 625/625 [==============================] - 381s 609ms/step - loss: 0.1862 - accuracy: 0.9312 - val_loss: 0.2697 - val_accuracy: 0.8877

Accuracy: 88.77%
```

6. Составим модель 2. Добавим одномерный слой CNN и максимальный пул после входного слоя.

```
model = Sequential()
model.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
```

7. Модель 2 достигла точности 90% на тестовых данных.

```
Epoch 1/3
625/625 [==============================] - 108s 171ms/step - loss: 0.5038 - accuracy: 0.7165 - val_loss: 0.2954 - val_accuracy: 0.8754
Epoch 2/3
625/625 [==============================] - 111s 178ms/step - loss: 0.2654 - accuracy: 0.8935 - val_loss: 0.2708 - val_accuracy: 0.9012
Epoch 3/3
625/625 [==============================] - 113s 180ms/step - loss: 0.1704 - accuracy: 0.9401 - val_loss: 0.2462 - val_accuracy: 0.9010
Accuracy: 90.10%
```

8. Составим модель 3. Добавим к 2 модели два слоя Dropout с вероятностью исключения 20%.

```
model = Sequential()
model.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
model.add(LSTM(100, return_sequences=True, dropout=0.3))
model.add(LSTM(50))
model.add(Dense(1, activation='sigmoid'))
```

9. Модель 3 достигла точности 96%

```
Epoch 1/3
```

```
625/625    [==============================]    -    139s
221ms/step - loss: 0.4992 - accuracy: 0.7226 - val_loss:
0.2177 - val_accuracy: 0.9190
Epoch 2/3
625/625    [==============================]    -    150s
241ms/step - loss: 0.2216 - accuracy: 0.9169 - val_loss:
0.1858 - val_accuracy: 0.9388
Epoch 3/3
625/625    [==============================]    -    142s
227ms/step - loss: 0.1684 - accuracy: 0.9390 - val_loss:
0.1332 - val_accuracy: 0.9612
```

10. Составим ансамбль из 4 моделей №3 с разными данными.

```
for              i              in              range(m):
    x_train  =  X_train[int(len(X_train)  /  m)  *  i:
int(len(X_train)     /     m)     *     (i     +     1)]
    y_train  =  Y_train[int(len(Y_train)  /  m)  *  i:
int(len(Y_train)     /     m)     *     (i     +     1)]

    models.append(Sequential())
    models[i].add(Embedding(top_words,
embedding_vector_length,
input_length=max_review_length))
    models[i].add(Conv1D(filters=32,      kernel_size=3,
padding='same',                   activation='relu'))
    models[i].add(MaxPooling1D(pool_size=2))
    models[i].add(Dropout(0.2))
    models[i].add(LSTM(100))
    models[i].add(Dropout(0.2))
    models[i].add(Dense(1,        activation='sigmoid'))

    models[i].compile(loss='binary_crossentropy',
optimizer='adam',                metrics=['accuracy'])
    models[i].fit(x_train,                       y_train,
validation_data=(x_train,       y_train),       epochs=3,
batch_size=64)
    print(models[i].evaluate(x_train,            y_train,
verbose=0))
    scores.append(models[i].evaluate(x_train,    y_train,
verbose=0)[1])

print("Accuracy: %.2f%%" % (np.mean(scores)*100))
```

11. Модель показала среднюю точность 97.09%

```
Epoch 1/3

157/157     [==============================]     -     42s
259ms/step - loss: 0.6849 - accuracy: 0.5350 - val_loss:
0.3984 - val_accuracy: 0.8248

Epoch 2/3

157/157     [==============================]     -     46s
292ms/step - loss: 0.3575 - accuracy: 0.8537 - val_loss:
0.2267 - val_accuracy: 0.9214

Epoch 3/3

157/157     [==============================]     -     44s
283ms/step - loss: 0.1970 - accuracy: 0.9269 - val_loss:
0.1188 - val_accuracy: 0.9671

[0.11883026361465454, 0.9671000242233276]

Epoch 1/3

157/157     [==============================]     -     45s
275ms/step - loss: 0.6829 - accuracy: 0.5362 - val_loss:
0.3052 - val_accuracy: 0.8779

Epoch 2/3

157/157     [==============================]     -     45s
289ms/step - loss: 0.3072 - accuracy: 0.8765 - val_loss:
```

```
0.2296 - val_accuracy: 0.9148

Epoch 3/3

157/157     [===============================]     -     46s
295ms/step - loss: 0.1794 - accuracy: 0.9374 - val_loss:
0.0814 - val_accuracy: 0.9803

[0.08135703951120377, 0.9803000092506409]

Epoch 1/3

157/157     [===============================]     -     50s
305ms/step - loss: 0.6780 - accuracy: 0.5522 - val_loss:
0.3304 - val_accuracy: 0.8647

Epoch 2/3

157/157     [===============================]     -     45s
290ms/step - loss: 0.3079 - accuracy: 0.8726 - val_loss:
0.1629 - val_accuracy: 0.9526

Epoch 3/3

157/157     [===============================]     -     46s
294ms/step - loss: 0.1597 - accuracy: 0.9483 - val_loss:
0.0870 - val_accuracy: 0.9769

[0.08702462911605835, 0.9768999814987183]

Epoch 1/3

157/157     [===============================]     -     49s
303ms/step - loss: 0.6636 - accuracy: 0.5730 - val_loss:
0.5680 - val_accuracy: 0.7165

Epoch 2/3

157/157     [===============================]     -     47s
301ms/step - loss: 0.4444 - accuracy: 0.8106 - val_loss:
0.1797 - val_accuracy: 0.9441
```

```
Epoch 3/3
157/157 [==============================] - 45s 287ms/step - loss: 0.1833 - accuracy: 0.9330 - val_loss: 0.1198 - val_accuracy: 0.9593
[0.11981038749217987, 0.9592999815940857]
Accuracy: 97.09%
```

12. Была написана функция для тестирования модели на пользовательском тексте.
13. Модель 3 была протестирована на 4 текстах. Они представлены в приложении А. Результаты тестирования:

Accuracy: 97.59%

0.78498954

It is positive feedback

Answer is true.

0.03490123

It is negative feedback

Answer is true.

0.12401813

It is negative feedback

Answer is true.

```
0.98995364

It is positive feedback

Answer is true.
```

Вывод.

В ходе лабораторной работы была построена нейронная сеть, для классификации обзоров на фильмы. Была достигнута точность модели 97%. Был разработан функционал, позволяющий считывать пользовательский текст из файла. Сеть была протестирована на пользовательском тексте обзора.

Приложение А.

Текст рецензий.

Рецензия 1.

Позитивная рецензия.

The film is simply gorgeous! Best movie I've ever seen. Everything is great. And the script, and directing, and special effects, well, the music… Hans Zimmer did his best (standing ovation)! The actors also played well… Everything is gorgeous. The only thing Nolan picked up a little bit… I immediately noticed this (although of course it is clear that this was done for the "piquancy" of the script and the logic of what is happening)... In a dream, time goes not slower, but faster than in reality. Well, for example. How much do you sleep at night?... Maybe 7-8 hours?… But you do not see a dream for 15-16 hours, but only a few minutes (usually 2-3 minutes), or even less than a minute! In the film, the hero DiCaprio explains to the girl that 5 minutes of sleep in reality is equal to several hours in a dream! And, consequently, the more levels of sleep, the slower time goes (who watched-will understand what I mean)! It's almost like this… When the hero DiCaprio with his beloved creates more and more levels of sleep, then in the end they would spend together not 50 years (!), as they say in the film, but only a few seconds (!). But I understand Nolan. The logic of the script and all that… I'd do that too… A little more than that…

Рецензия 2.

Негативная рецензия.

This is the worst movie I've ever seen. Without exception, all the actors played poorly.

Рецензия 3.

Негативная рецензия.

"The budget is visible in everything, and this concerns artistic implementation, not technical capabilities. The film certainly does not cause wild disgust, it is quite watchable, dynamic, with more or less lively actors (who really play, and do not utter pathetic speeches with a fixed expression), with streams of blood and what is most interesting, comical and parodic. And at the same time, it is absolutely not a successful film adaptation. The most important element is missing, without which everything is meaningless - there is no tournament, there are just chaotic clashes of opponents (although the existence of the outer and earthly worlds is described in detail and the rules of the game are announced) and there is no convincing atmosphere and faith in the reality of what is happening (which was actually famous for the old dilogy). A monotonous cold color, no colors and everything faded, terrible heavy costumes, some "garbage" scenery-the action is in barns, in some mines, or even in some landfill.

Рецензия 4.

Положительная рецензия.

Recently, while typing in a search engine something like "existential comedies", I came across the series "High with delivery". Despite the high rating among film critics, it is not at all popular with the Russian-speaking audience. Maybe because of the translation (it sounds like another stupid comedy), or maybe because of the fact that viewing requires some tolerance, liberality in views (n-r, on the use of marijuana). But anyway, the series turned out to be absolutely wonderful. The plot is simple — the main character," the guy with the weed " delivers marijuana to customers, each episode is the story of a new client.
The movie is both sad and funny, without complicated ideas and pretentious messages. About ordinary people and about life that just

happens. A kaleidoscope of life stories. The main character is a very real character (he, by the way, is played by the director himself), well, just a guy from the next street. He often gets into a little ridiculous situations — you look and do not know how to react: laughter and sadness, sympathy and Spanish shame — and often all at the same time. And his bike rides around New York City to music are a meditation that deserves special attention.

I wonder why I was so hooked on the show? First of all, the mood. There is something Japanese about it, some elusive zen, a reminder-from which you want to stop, cancel all urgent matters and go for a walk in the nearest park, enjoy nature, a walk and pleasant company-simple joys. I like that the author does not have a clear message for the viewer, I think everyone will have their own message. For me, this is about celebrating life, about the versatility of life stories, about beauty in the ordinary.