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

Студент гр. 8383 _____ Ларин А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

Классификация последовательностей - это проблема прогнозирующего моделирования, когда у вас есть некоторая последовательность входных данных в пространстве или времени, и задача состоит в том, чтобы предсказать категорию для последовательности.
Проблема усложняется тем, что последовательности могут различаться по длине, состоять из очень большого словарного запаса входных символов и могут потребовать от модели изучения долгосрочного контекста или зависимостей между символами во входной последовательности.
В данной лабораторной работе также будет использоваться датасет IMDb, однако обучение будет проводиться с помощью рекуррентной нейронной сети.

Задание

- Ознакомиться с рекуррентными нейронными сетями
- Изучить способы классификации текста
- Ознакомиться с ансамблированием сетей
- 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)

Выполнение

В работе требуется классифицировать обзоры. В работе используетсся датасет imdb

В начале импортируются все необходиные зависимости включая датасет imdb, входящий в keras

```
import numpy as np
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
from keras.layers.convolutional import Conv1D, MaxPooling1D
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
```

Затем происходит загрузка и конкатенация датасета, а также его обрезка/дополнение до длины в 500 слов

```
top_words = 10000
(X_train, Y_train), (X_test, Y_test) = imdb.load_data(num_words=top_words)
data = np.concatenate((X_train, X_test), axis=0)
targets = np.concatenate((Y_train, Y_test), axis=0)

max_review_length = 500
data = sequence.pad_sequences(data, maxlen=max_review_length)
```

Для начала создана простая модель из методических указаний. Она имеет следующий вид:

```
model = Sequential()
model.add(Embedding(top_words, 32, input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
```

Данная модель использует слои LSTM т. н. долгой краткосрочной памяти. Они хранят информацию о предыдущих шагах. Они хорошо подходят для анализа последовательностей, таких как временные ряды или естественный язык.

Для отображения обзоров в векторную область используется техника встраивания(embedding). Каждое слово сопоставляется с вектором определенной длины

Модель скомпилирована и обучена на трех эпохах при размере батча 64 и биением данных на тренировочные и валидационные в отношении 8:2

```
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
print(model.summary())
H = model.fit(data, targets, validation_split=.2, epochs=3, batch_size=64)
```

По результатам была получена точность 0.8875 (при валидации)

Далее был добавлен слой свертки после векторизации, и слой пуллинга после него.

```
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_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'))
```

По результатам точность составила 0.8873 т. е. почти не изменилась, но время обучения снизилось(43 сек. против 61 у пред. модели)

Рекурентные сети склонны к переобучению. Чтобы этого не случалось добавлены слои dropout

```
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_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(Dropout(0.3))
model.add(LSTM(100))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
```

Была получена точность 0.9027

Для повышения точности было переменено ансамблирование моделей. Обучено несколько моделей на разных данных. Результат работы ансамбля считается как среднее от всех результатов

Было обучено 5 моделей. В конце обучения модели сохраняются в файл model#.h5

Затем был написан модуль для использования моделей на пользовательских данных. Он принимает на вход(в аргументах командной строки) путь к файлу, читает из него отзыв, прогоняет через модели, считает среднее и печатает результат на экран

Результаты тестирования:
1.
Входные данные:
Good stuff. Love it! Best thing ever! This film is amazing.
Ответ модели:
[0.753990888595581, 0.8335102796554565, 0.7318012714385986,
0.7261291146278381, 0.6374814510345459]
Среднее:[[0.73658264]]

2.
Входные данные:
Very bad film. Didn't like it
Ответ модели:
[0.3881857991218567, 0.3244476318359375, 0.313212513923645,
0.3827672302722931, 0.4140523672103882]
Среднее:[[0.36453313]]

Видно, что отдельные модели могут довольно сильно отклоняться от верного ответа. При меньшей степени уверенности можно ожидать что в среднем ансамбль даст правильный ответ не смотря на ошибки отдельных моделей

Выводы.

Были изучен способы векторизированного представления естественного языка для дальнейшей обработки . Были изучены рекурентные нейронные сети(LSTM). На основе датасета imdb обучена модель для классификации отзывов на фильмы. По результатам получена точность 0.9027

Был построен ансамбль моделей, способный дать более надежную оценку, чем отдельные модели в его составе.
Был написан модуль для прогона модели по пользовательским данным. С его помощью ансамбль была протестирована. Для положительного отзыва получен ответ 0.73658264 для отрицательного 0.36453313. Результаты тестов удовлетворительны