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. 7381 Процветкина А.В.

Преподаватель Жукова Н.А.

Санкт-Петербург

2020

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.

Постановка задачи.

- Ознакомиться с рекуррентными нейронными сетями
- Изучить способы классификации текста
- Ознакомиться с ансамблированием сетей
- Построить ансамбль сетей, который позволит получать точность не менее 97%

Ход работы.

Каждому обзору фильма сопоставим вещественное число. Слова кодируются как действительные векторы в многомерном пространстве, где семантическое сходство между словами приводит к близости в векторном пространстве.

Keras предоставляет удобный способ преобразования положительных целочисленных представлений слов в вложение слов слоем Embedding.

Общее количество слов, интересных в моделировании, было ограничено до 5000 наиболее часто встречающихся слов. Длина последовательности (количество слов) в каждом обзоре варьируется, поэтому каждый обзор был

ограничен 500 словами, укорачивая длинные обзоры и дополняя короткие обзоры нулевыми значениям.

Сначала были созданы и сконфигурированы две различные НС. Первая: LSTM, вторая: CNN+LSTM.

Обе модели были обучены на датасете IMDb на 3х эпохах.

Из моделей был сформирован ансамбль с целью увеличения точности предсказаний. При создании ансамбля подбирались веса для предсказаний отдельных моделей: (0.2, 0.8), (0.3, 0.7), (0.5, 0.5). (Вес для LSTM-модели ниже во всех случаях по причине того, что точность этой модели в принципе ниже.) Точность отдельных моделей и точность ансамбля предоставлены на рис.1.

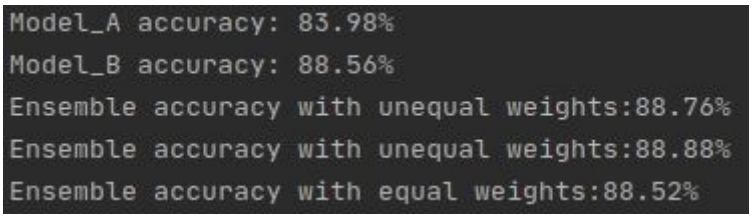

Рисунок 1 -- Точности моделей и ансамблей

Можно заметить, что оптимальными оказались веса (0.3, 0.7) и точность модели пусть и не намного, но увеличилась.

Была написана функция `ensemble_prediction(models, weights, X)`, позволяющая предсказать окраску пользовательского отзыва с помощью ансамбля. Исходный код предоставлен в приложении А, а примеры отзывов в приложениях Б и В.

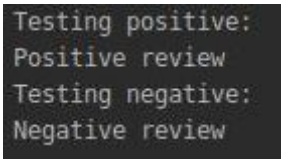

Рисунок 2 -- Результат предсказаний ансамбля

Выводы.

В ходе лабораторной работы была изучена задача ансамблирования моделей, построена и обучена ИНС на основе датасета IMDB. Ансамблированная модель была проверена на пользовательском отзыве. Точность модели в результате обучения составила 88.88%, в то время как точность отдельных моделей оказалась равна 83.98% и 88.56%.

Приложение А

Исходный код программы

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

top_words = 5000
(X_train, y_train), (X_test, y_test) =
imdb.load_data(num_words=top_words)
# truncate and pad input sequences
max_review_length = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)
# create the model
embedding_vector_length = 32
model_A = Sequential()
model_A.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
model_A.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model_A.add(Dense(1, activation='sigmoid'))
model_A.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
print("Start fitting A")
model_A.fit(X_train, y_train, epochs=3, batch_size=64)

print("Start evaluating A")
scores_A = model_A.evaluate(X_test, y_test, verbose=0)

model_B = Sequential()
model_B.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
model_B.add(Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'))
model_B.add(MaxPooling1D(pool_size=2))
model_B.add(LSTM(100))
model_B.add(Dense(1, activation='sigmoid'))
model_B.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
print("Start fitting B")
model_B.fit(X_train, y_train, epochs=3, batch_size=64)
```

```
print("Start evaluating B")
scores_B = model_B.evaluate(X_test, y_test, verbose=0)

print("Model_A accuracy: %.2f%%" % (scores_A[1]*100))
print("Model_B accuracy: %.2f%%" % (scores_B[1]*100))

def ensemble_prediction(models, weights, X):
    yhats = [model.predict(X) for model in models]
    yhats = np.array(yhats)
    weighted_res = np.tensordot(yhats, weights, axes=((0),(0)))
    return np.rint(weighted_res)


models = [model_A, model_B]

weights = [0.2, 0.8]
y_hat_not_eq = ensemble_prediction(models, weights, X_test)
ACC = accuracy_score(y_test, y_hat_not_eq)
print("Ensemble accuracy with unequal weights:%.2f%%" % (ACC*100))

weights = [0.3, 0.7]
y_hat_not_eq = ensemble_prediction(models, weights, X_test)
ACC = accuracy_score(y_test, y_hat_not_eq)
print("Ensemble accuracy with unequal weights:%.2f%%" % (ACC*100))

weights = [0.5, 0.5]
y_hat_eq = ensemble_prediction(models, weights, X_test)
ACC = accuracy_score(y_test, y_hat_eq)
print("Ensemble accuracy with equal weights:%.2f%%" % (ACC*100))


def vectorize(sequences, dimension=top_words):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results


def custom_review(name):
    f = open(name, 'r')
    f_text = f.read()
    index = imdb.get_word_index()
    txt = []
    for i in f_text:
        if i in index and index[i] < top_words:
            txt.append(index[i])

    txt = vectorize([txt])
    return txt
```

```
weights = [0.3, 0.7]
text = custom_review("pos_review.txt")
text = sequence.pad_sequences(text, maxlen=max_review_length)
print("Testing positive:")
result = ensemble_prediction(models, weights, text)
if result == 1:
    print("Positive review")
else:
    print("Negative review")

print("Testing negative:")
text = custom_review("neg_review.txt")
text = sequence.pad_sequences(text, maxlen=max_review_length)
result = ensemble_prediction(models, weights, text)

if result == 1:
    print("Positive review")
else:
    print("Negative review")
```

Приложение Б

Положительный отзыв

How often do you see a movie like this? It absolutely rocks. Even though it tones down the violence of the dark horse comics, it seamlessly blends Tex Avery style wackiness into the real world. If that does'nt reflect the inner child in us and what everyone wants to be, then what does? Firstly, the main character is brilliant. Hes an explosive combination of all the wacky toons we all love. But the difference is that its all amplified 10 times over and its all real! You actually combine all this wackiness into a very serious, realistic, dark and gritty world. The guy who came up with the 'Tornado' routine needs to be given a major award. This has to be Jim Carrey's best performance in his career and for once, hes not overacting or being annoying. Cameron Diaz looks her best in this movie.

Приложение В

Негативный отзыв

This is hilarious! I cannot believe I have been waiting for this movie so long to just get disappointed. It looks like no one makes good horrors anymore. Why making main characters such idiots? Moreover actors look like they understand they're playing idiots. Wanted to start yawning after an hour of watch. Boring. I regret spending money on a such rubbish.