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 _____ Мололкин К.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.

Задачи

- 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 работы

В начале работы были загружены и векторизованы данные IMDB. Затем были построена модель нейронной сети со следующей структурой, представленной на рис. 1.

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

Рисунок 1 – первая модель

Данная модель показывает точность: 90% для обучающих данных и 80% для тестовых.

Затем первая модель была изменена, структура новой модели представлена на рис. 2.

```
model2 = Sequential()
model2.add(Embedding(max_words, embedding_vector_length, input_length=max_review_length))
model2.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model2.add(MaxPooling1D(pool_size=2))
model2.add(LSTM(100))
model2.add(Dense(1, activation='sigmoid'))
```

Рисунок 2 – вторая модель

Данная модель показывает точность: 92% для обучающих данных и 87% для тестовых.

Во вторую модель были добавлены два слоя Dropout, данная модель представлена на рис. 3.

```
model3 = Sequential()
model3.add(Embedding(max_words, embedding_vector_length, input_length=max_review_length))
model3.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model3.add(MaxPooling1D(pool_size=2))
model3.add(Dropout(0.2))
model3.add(LSTM(100))
model3.add(Dropout(0.2))
model3.add(Dense(1, activation='sigmoid'))
```

Данная модель показывает точность: 95% для обучающих данных и 90% для тестовых.

Так как 3 модель показывает лучший результат, она и будет использована для ансамблирования. Точность данной модели – 98.15%.

Так же были выбраны 2 реальных отзыва:

1. Finally something new and fresh from the stereotypical drivel being jammed down our throats for the past couple of years. None of the clichés either. Dark,gory hilarious, ethically mind bending. Like how far would you twist justice to serve your needs, what price would you be willing to pay for revenge is it worth losing your humanity? All this mixed into a beautiful first season. Can't wait for more. A definite must watch epically if you enjoyed titles like Titans.

   Результат: 99.85%
2. This was a great show. Season 1 was fantastic. My assumption is they hired a new woke writers room in season 2, who only took a few episodes to shoe-horn in the typical preaching agenda in the most on the nose way I've ever seen. You can tell how forced it is and that it wasn't the initial trajectory of the show. Incredibly disappointing, but unfortunately expected these days. It seems anything good picks up larger producers who enforce diversity requirements on writers rooms and then it's only a matter of time until we end up with another stale and cringy show awash with tired political tropes.

   Результат: 7.34%

Вывод

В ходе выполнения лабораторной работы были изучены рекуррентные нейронные сети, способы классификации текста, способ создания ансамбля моделей. А также построена нейронная сеть, классифицирующая обзоры фильмов по отзывам с IMDB.

Приложение А

Листинг программы

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

def get_vec_from_file(file):
    with open(file, 'r') as txtFile:
        text = txtFile.read()
    words = text_to_word_sequence(text)
    vocabulary = imdb.get_word_index()
    x_predict = []
    for i in range(len(words)):
        if words[i] not in vocabulary:
            continue
        if vocabulary[words[i]] + 3 < 10000:
            x_predict.append(vocabulary[words[i]] + 3)
    return sequence.pad_sequences([x_predict], maxlen=max_review_length)


def predict_sample_text(pred):
    predictions = []
    for i in range(m):
        predictions.append(models[i].predict(pred))
    prediction = ((np.mean(predictions)) > 0.5).astype("int32")
    print("Mean predictions %.2f%%" % (np.mean(predictions) * 100))
    if prediction == 1:
        print("Good")
    else:
        print("Bad")

max_words = 10000
(training_data, training_targets), (testing_data, testing_targets) =
      imdb.load_data(num_words=max_words)
train_x = np.concatenate((training_data, testing_data), axis=0)
train_y = np.concatenate((training_targets, testing_targets), axis=0)
train_y = np.array(train_y).astype("float32")

max_review_length = 500
train_x = sequence.pad_sequences(train_x, maxlen=max_review_length)
embedding_vector_length = 32

model1 = Sequential()
model1.add(Embedding(max_words, embedding_vector_length,
      input_length=max_review_length))
model1.add(LSTM(100))
model1.add(Dense(1, activation='sigmoid'))

model2 = Sequential()
model2.add(Embedding(max_words, embedding_vector_length,
      input_length=max_review_length))
model2.add(Conv1D(filters=32, kernel_size=3, padding='same',
      activation='relu'))
model2.add(MaxPooling1D(pool_size=2))
model2.add(LSTM(100))
```

```
model2.add(Dense(1, activation='sigmoid'))

model3 = Sequential()
model3.add(Embedding(max_words, embedding_vector_length,
      input_length=max_review_length))
model3.add(Conv1D(filters=32, kernel_size=3, padding='same',
      activation='relu'))
model3.add(MaxPooling1D(pool_size=2))
model3.add(Dropout(0.2))
model3.add(LSTM(100))
model3.add(Dropout(0.2))
model3.add(Dense(1, activation='sigmoid'))

models = []
scores = []
m = 4
for i in range(m):
    x_train = train_x[int(len(train_x) / m) * i: int(len(train_x) / m) * (i +
      1)]
    y_train = train_y[int(len(train_y) / m) * i: int(len(train_y) / m) * (i +
      1)]

    models.append(model3)
    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("Mean accuracy: %.2f%%" % (np.mean(scores) * 100))

while True:
    print("Input filename with review or 0 to stop")
    filename = input()
    if filename != "0":
        predict_sample_text(get_vec_from_file(filename))
    else:
        break
```