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	___________________	Дейнега В.Е.
Преподаватель	___________________	Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

Прогноз успеха фильмов по обзорам (Predict Sentiment From Movie Reviews).

Задание.

-Ознакомиться с рекуррентными нейронными сетями

-Изучить способы классификации текста

-Ознакомиться с ансамблированием сетейНайти набор оптимальных ИНС для классификации текста

-Провести ансамблирование моделей

-Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей

-Провести тестирование сетей на своих текстах (привести в отчете)

Выполнение работы

1) Создадим модель ИНС.

```
        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(0.5))
        model.add(LSTM(20, return_sequences=True, dropout=0.5))
        model.add(Dropout(0.5))
        model.add(LSTM(30))
        model.add(Dense(1, activation='sigmoid'))
        model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])
        model.fit(partial_train_data,partial_train_targets,
validation_data=(val_data, val_targets),epochs=3, batch_size=64, )
```

Модель обучается на 3-х эпохах из-за дальнейшего переобучения, эту проблему не удалось преодолеть добавлениями слоев dropout.

2) Повысим точность модели, воспользовавшись cross-validated committees по k фолдам. Обучим 5 моделей ИНС на разных обучающих и тестовых наборах.

```
for i in range(num_models):
    val_data = data[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = targets[i * num_val_samples: (i + 1) * num_val_samples]
    tmp = data[(i + 1) * num_val_samples:]
    partial_train_data = np.concatenate([data[:i * num_val_samples],
           data[(i+1)* num_val_samples:]], axis=0)
    partial_train_targets=np.concatenate([targets[:i* num_val_samples],
           targets[(i+1)*num_val_samples:]], axis=0)
    partial_train_data=sequence.pad_sequences(partial_train_data,
maxlen=max_review_length)
    val_data=sequence.pad_sequences(val_data, maxlen=max_review_length)

    ##create model##

    models.append(model)
    model_json = model.to_json()
    with open(str(i) + '.json', "w") as json_file:
        json_file.write(model_json)
    model.save_weights(str(i) + '.h5')
```

3) Была написана функция пользовательского ввода текста из файла. Пользователь вводит название файла с текстом рецензии.

```
def user_input():
    print("Type file name:\n")
    filename = input()
    f = open(filename+".txt", "r")
    text = f.read()
    prediction(text_prepare(text))
    return 0
```

Вспомогательная функция, подготавливающая и кодирующая текст.

```
def text_prepare(text):
    text = text.lower().strip()
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'\n', ' ', text)
    text = text.split(' ')
    index = imdb.get_word_index()
    coded = []

    for word in text:
        ind = index.get(word)
        if ind is not None and ind < top_words:
    coded = np.array([coded])
    return sequence.pad_sequences(coded, maxlen=max_review_length)
```

Протестируем нейросеть, на обзорах фильмов Yesterday (2019) и The Room (2003), оценка рецензии ставится как среднее значение оценок каждой модели.

Текст рецензии Yesterday (2019):

“I went into this wondering if they were going to explain the whole premise with it being a dream or such? Was he going to get caught and crucified for it? Was he going to get the girl, lose the girl? So many movies are fixated on realism and difficulties and negativity. Yes, he faces several difficulties. But, the movie is light and beautiful with fantastic music. If I want realism, I watch a documentary. This is what I go to the movies for. Really well done!”

```
0.9854601979255676
Good
```

Текст рецензии The Room (2003):

“I've never in my life been more entertained by a film that has absolutely NO redeeming qualities. Unintentionally inept characters engage in progressively bizarre and unnatural interactions which seem to peak at erratic and unexpected intervals. The awkwardness of the actors is framed by strange pauses, jarring scripts and incredibly bizarre production techniques - there are ample 'deer in the headlights' moments, in which you can feel genuine sympathy for these people who are obviously so caught up in Tommy's strange and dominating creative control that they've failed to see any better.

Other filmmakers play with similarly surreal concepts - David Lynch for example - but this film lacks anything resembling artistic refinement, insight or self awareness placing it far from comparison. It's kind of like watching a train crash in slow motion - random, incoherent, disastrous, accidental and ultimately painful. The sense of alienation emanating from this film places the audience extremely far from being able to relate to what's happening on screen, which leaves a lot of room for uncontrollable laughter given the right circumstances.

The camera work and production techniques would not be out of place in many daytime soap operas, nor would the script and plot, but there is an undefinable quality which separates this movie from the sense mediocrity often found in such shows and instead casts it deep into the abyss of tragically bad film making where it will be forever trapped along with Wiseau's artistic integrity. This really is a new frontier.

It is truly awful, but I cannot recommend it enough."

```
0.00604676604270935
Bad
```

Выводы.

В ходе лабораторной работы был реализован прогноз успеха фильмов по обзорам. Также был применен метод ансамблирования моделей для более точного семантического анализа текста.

Приложение A

```
	import numpy as np
	from keras.datasets import imdb
	from keras.models import Sequential
	from keras.layers import Dense
	from keras.layers import Conv1D
	from keras.layers import MaxPooling1D
	from keras.layers import LSTM
	from keras.layers import Dropout
	from keras.layers.embeddings import Embedding
	from keras.preprocessing import sequence
	import tensorflow as tf
	from keras.datasets import imdb
	from keras.models import model_from_json
	import re

	#mpl.use('TKAgg')
	top_words = 10000
	(training_data,    training_targets),    (testing_data,    testing_targets)    =
imdb.load_data(num_words=top_words)
	data = np.concatenate((training_data, testing_data), axis=0)
	targets = np.concatenate((training_targets, testing_targets), axis=0)
	targets = np.array(targets).astype("float32")


	num_models = 5
	max_review_length = 500
	num_val_samples = len(data) // num_models
	embedding_vecor_length = 32

	models = []

	# for i in range(num_models):
	#     val_data = data[i * num_val_samples: (i + 1) * num_val_samples]
	#     val_targets = targets[i * num_val_samples: (i + 1) * num_val_samples]
	#     tmp = data[(i + 1) * num_val_samples:]
	#     partial_train_data = np.concatenate([data[:i * num_val_samples],
	#                                          data[(i + 1) * num_val_samples:]],
axis=0)
	#     partial_train_targets = np.concatenate([targets[:i * num_val_samples],
	#                                             targets[(i + 1) * num_val_samples:]],
axis=0)
	#
	#              partial_train_data    =    sequence.pad_sequences(partial_train_data,
maxlen=max_review_length)
	#     val_data = sequence.pad_sequences(val_data, maxlen=max_review_length)
	#
	#     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(0.5))
	#     model.add(LSTM(20, return_sequences=True, dropout=0.5))
	#     model.add(Dropout(0.5))
	#     model.add(LSTM(30))
	#     model.add(Dense(1, activation='sigmoid'))
	#               model.compile(loss='binary_crossentropy',    optimizer='adam',
metrics=['accuracy'])
```

```
    #                        model.fit(partial_train_data,     partial_train_targets,
validation_data=(val_data, val_targets),
    #               epochs=3, batch_size=64, )
    #     models.append(model)
    #     model_json = model.to_json()
    #     with open(str(i) + '.json', "w") as json_file:
    #         json_file.write(model_json)
    #     model.save_weights(str(i) + '.h5')

    def load_model(str):
        json_file = open(str + '.json', 'r')
        loaded_model_json = json_file.read()
        json_file.close()
        loaded_model = model_from_json(loaded_model_json)
        loaded_model.load_weights(str + '.h5')
        loaded_model.compile(loss='binary_crossentropy',            optimizer='adam',
metrics=['accuracy'])
        return loaded_model


    for i in range(num_models):
        model = load_model(str(i))
        models.append(model)


    def text_prepare(text):
        text = text.lower().strip()
        text = re.sub(r'[^\w\s]', '', text)
        text = re.sub(r'\n', ' ', text)
        text = text.split(' ')
        index = imdb.get_word_index()
        coded = []

        for word in text:
            ind = index.get(word)
            if ind is not None and ind < top_words:
                coded.append(ind + 3)
        coded = np.array([coded])
        return sequence.pad_sequences(coded, maxlen=max_review_length)


    def user_input():
        print("Type file name:\n")
        filename = input()
        f = open(filename+".txt", "r")
        text = f.read()
        prediction(text_prepare(text))
        return 0


    def prediction(coded):
        res = np.zeros(num_models)
        for i in range(num_models):
            res[i] = models[i].predict(coded)

        mean_res = np.mean(res)
        print(mean_res)
        if mean_res >= 0.5:
            print('good')
        else:
            print('bad')
```

```
user_input()
```