МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Искусственные нейронные сети»

Тема: Прогноз успеха фильмов по обзорам

Студент гр.8382 _____ Терехов А.Е.

Преподаватель _____ Жангриров Т.Р.

Санкт-Петербург

2021

Цель работы

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

Задачи

- Ознакомиться с задачей классификации.
- Изучить способы представления текста для передачи в ИНС.
- Достигнуть точность прогноза не менее 95%.

Требования

1. Построить и обучить нейронную сеть для обработки текста.
2. Исследовать результаты при различном размере вектора представления текста.
3. Написать функцию, которая позволяет ввести пользовательский текст (в отчете привести пример работы сети на пользовательском тексте).

Основные теоретические положения

Что такое анализ настроений (сентимент-анализ)?

С помощью анализа настроений можно определить отношение (например, настроение) человека к тексту, взаимодействию или событию. Поэтому сентимент-анализ относится к области обработки естественного языка, в которой смысл текста должен быть расшифрован для извлечения из него тональности и настроений.

Спектр настроений обычно подразделяется на положительные, отрицательные и нейтральные категории. С использованием анализа настроений можно, например, прогнозировать мнение клиентов и их отношение к продукту на основе написанных ими обзоров. Поэтому анализ настроений широко

применяется к обзорам, опросам, текстам и многому другому.

Датасет IMDb состоит из 50 000 обзоров фильмов от пользователей, помеченных как положительные (1) и отрицательные (0).

- Рецензии предварительно обрабатываются, и каждая из них кодируется последовательностью индексов слов в виде целых чисел.
- Слова в обзорах индексируются по их общей частоте появления в датасете. Например, целое число «2» кодирует второе наиболее частое используемое слово.
- 50 000 обзоров разделены на два набора: 25 000 для обучения и 25 000 для тестирования.

Датасет был создан исследователями Стэнфордского университета и представлен в статье 2011 года, в котором достигнутая точность предсказаний была равна 88,89%. Датасет также использовался в рамках конкурса сообщества Kaggle «Bag of Words Meets Bags of Popcorn» в 2011 году.

Ход работы

В работе были использованы следующие зависимости:

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
from keras import layers
from keras.datasets import imdb
from keras.models import Sequential
from sklearn.model_selection import train_test_split
```

Загрузка и обработка датасета:

```
(training_data, training_targets), (testing_data, testing_targets
   ) = imdb.load_data(
```

```
        num_words=NUM_WORDS + SKIP_WORDS, skip_top=SKIP_WORDS)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets),
   axis=0)
data = vectorize(data)
targets = np.array(targets).astype("float32")
x_train, x_test, y_train, y_test = train_test_split(data, targets
   , test_size=0.2)
```

Функция для векторизации данных:

```
def vectorize(sequences:List[List[int]], dimension:int=NUM_WORDS)
    -> np.ndarray:
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results
```

Так как отзывы представлены в числовом закодированном формате, для расшифровки отзыва была написана функция:

```
def decode_report(report: List[int])->str:
    index = imdb.get_word_index()
    reverse_index = dict([(value, key) for (key, value) in index.
       items()])
    decoded = " ".join([reverse_index.get(i - 3, "#") for i in
       report])
    return decoded
```

Пример декодированного текста.

```
french horror cinema has seen something of a revival over the
   last couple of years with great films such as inside and #
   romance # on to the scene # # the revival just slightly but
   stands head and shoulders over most modern horror titles and
   is surely one of the best french horror films ever made # was
```

```
obviously shot on a low budget but this is made up for in far
more ways than one by the originality of the film and this in
turn is # by the excellent writing and acting that ensure the
film is a winner the plot focuses on two main ideas prison and
 black magic the central character is a man named # sent to
prison for fraud he is put in a cell with three others the
quietly insane # body building # marcus and his retarded
boyfriend daisy after a short while in the cell together they
stumble upon a hiding place in the wall that contains an old #
 after # part of it they soon realise its magical powers and
realise they may be able to use it to break through the prison
```

Можно заметить, что словарь IMDB хранит множество слов, но такие слова, как ’I’, ’and’, ’a’, ’the’, не несут важной информации, поэтому попробуем из исходного словаря удалить хотя бы 50 самых встречающихся субъективно бесполезных слов. Эти слова расположены в начале словаря, поэтому пропустим 50 слов в начале.

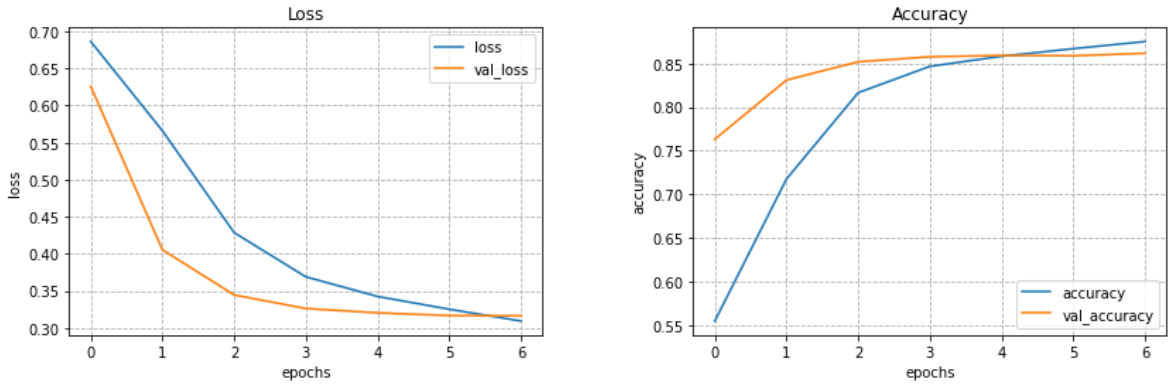
Для того чтобы построить модель, была написана функция.

```
def build_model() -> Sequential:
    model = Sequential()
    model.add(layers.Dense(100, activation="relu", input_shape=(
       NUM_WORDS,)))
    model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
    model.add(layers.Dense(100, activation="relu"))
    model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
    model.add(layers.Dense(50, activation="relu"))
    model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
    model.add(layers.Dense(1, activation="sigmoid"))
    return model
```

Построенная модель была скомпилирована и обучена при следующих параметрах.

```
model.compile(optimizer="adam", loss="binary_crossentropy",
   metrics=["accuracy"])
results = model.fit(x_train, y_train, epochs=2, batch_size=256,
   validation_data=(x_test, y_test))
```

При размере словаря 1000 слов были получены следующие результаты (рис.1).

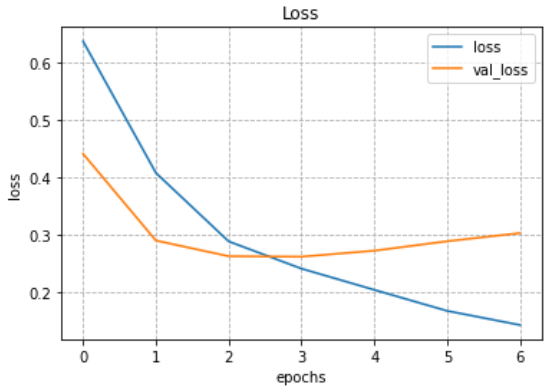


Потери Точность

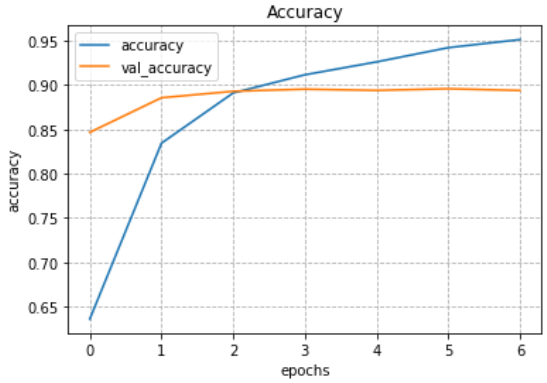
Рис. 1 – 1000 слов

За 7 эпох модель практически не переобучилась. Конечная точность составила 86%.

Увеличим размер словаря до 5000. Полученные результаты представлены на рисунке 2.



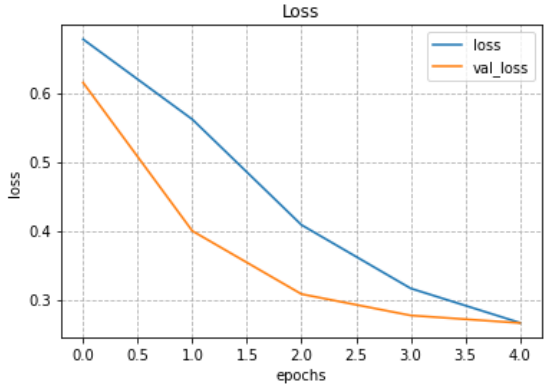
Потери



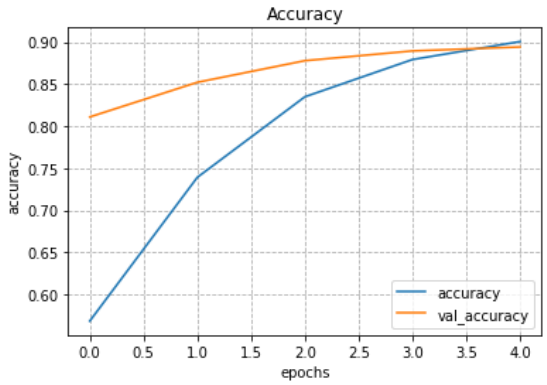
Точность

Рис. 2 – 5000 слов

За 7 эпох модель переобучилась, поэтому увеличим размер партии до 2^{12} и сократим число эпох до 5 (рис. 3).



Потери

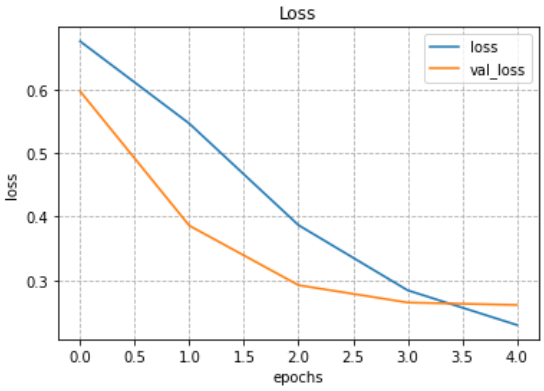


Точность

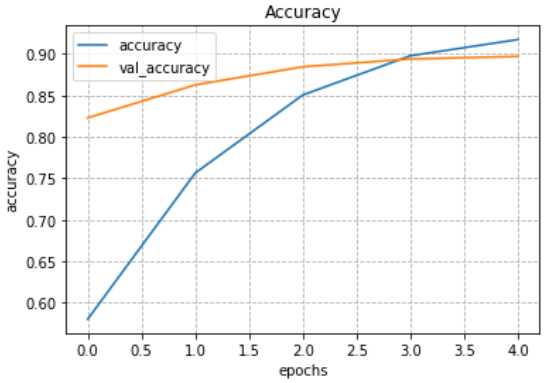
Рис. 3 – 5000 слов

За 5 эпох модель не успела переобучиться. Конечная точность на валидационных данных составила 89%.

Увеличим размер словаря до 10000.



Потери



Точность

Рис. 4 – 10000 слов

За 5 эпох модель незначительно переобучилась. Конечная точность на валидационных данных составила 89.7%.

Таким образом увеличение размера словаря положительно сказывается на точность модели, но в то же время модель становится более склонной к переобучению.

Для возможности загрузки пользовательских отзывов были написаны две функции.

```
def load_file(filename: str) -> str:
    with open(filename, "r") as f:
        return f.read()


def load_custom_text(text: str, dimension: int = NUM_WORDS) -> np
   .ndarray:
    text = text.lower()
    words = re.findall(r"\w+", text)
    index: dict = imdb.get_word_index()
    x = []
    for word in words:
        if word in index and index[word] < dimension:
```

```
            x.append(index[word])
    x = vectorize([np.array(x)])
    return x
```

Тексты отзывов были взяты с русскоязычного сайта и переведены. Окраска отзыва соответствует названию файла.

При обработке отзывов сетью были получены следующие результаты.

```
[[0.01934349]] Negative
[[0.7385045]] Positive
```

С данными отзывами сеть отработала корректно. Данные отзывы достаточно экспрессивные, то есть содержат много слов-маркеров, что также положительно сказалось на результатах.

Вывод

В ходе работы была изучена задача классификации настроений на примере отзывов о фильмах. Было изучено как передавать текстовые данные искусственной нейронной сети, как влияет размер словаря на результаты работы сети. Была получена сеть имеющая точность около 90%. Из эксперимента с двумя неизвестными для сети отзывами, следует, что сеть корректно определяет настроение текста.

ПРИЛОЖЕНИЕ А

```
import re
from typing import List

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from keras import layers
from keras.datasets import imdb
from keras.models import Sequential
from sklearn.model_selection import train_test_split

COLAB_PATH_PREFIX = "/content/drive/MyDrive/"
SKIP_WORDS = 50
NUM_WORDS = 10000


def plot(data: pd.DataFrame, label: str, title: str):
    axis = sns.lineplot(data=data, dashes=False)
    axis.set(ylabel=label, xlabel='epochs', title=title)
    axis.grid(True, linestyle="--")
    plt.show()


def decode_report(report: List[int]) -> str:
    index = imdb.get_word_index()
    reverse_index = dict([(value, key) for (key, value) in index.
       items()])
    decoded = " ".join([reverse_index.get(i - 3, "#") for i in
       report])
    return decoded
```

```
def vectorize(sequences: List[List[int]], dimension: int =
   NUM_WORDS) -> np.ndarray:
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, np.array(sequence) - SKIP_WORDS] = 1
    return results


def build_model() -> Sequential:
    model = Sequential()
    model.add(layers.Dense(100, activation="relu", input_shape=(
       NUM_WORDS,)))
    model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
    model.add(layers.Dense(100, activation="relu"))
    model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
    model.add(layers.Dense(50, activation="relu"))
    model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
    model.add(layers.Dense(1, activation="sigmoid"))
    return model


def save_sorted_index():
    index = imdb.get_word_index()
    sorted_index = dict(sorted(index.items(), key=lambda x: x[1])
       )
    with open("fout.csv", "w") as f:
        f.writelines([f"{i}, {w}\n" for w, i in sorted_index.
           items()])
```

```
def load_file(filename: str) -> str:
    with open(filename, "r") as f:
        return f.read()


def load_custom_text(text: str, dimension: int = NUM_WORDS) -> np
   .ndarray:
    text = text.lower()
    words = re.findall(r"\w+", text)
    index: dict = imdb.get_word_index()
    x = []
    for word in words:
        if word in index and index[word] < dimension:
            x.append(index[word])
    x = vectorize([np.array(x)])
    return x


if __name__ == '__main__':
    (training_data, training_targets), (testing_data,
       testing_targets) = imdb.load_data(
        num_words=NUM_WORDS + SKIP_WORDS, skip_top=SKIP_WORDS)
    data = np.concatenate((training_data, testing_data), axis=0)
    targets = np.concatenate((training_targets, testing_targets),
        axis=0)
    data = vectorize(data)
    print(data.shape)
    targets = np.array(targets).astype("float32")
    x_train, x_test, y_train, y_test = train_test_split(data,
       targets, test_size=0.2)
    model = build_model()
    model.compile(optimizer="adam", loss="binary_crossentropy",
```

```
    metrics=["accuracy"])
results = model.fit(x_train, y_train, epochs=5, batch_size
    =4096, validation_data=(x_test, y_test))
history = pd.DataFrame(results.history)
plot(history[['loss', 'val_loss']], 'loss', 'Loss')
plot(history[['accuracy', 'val_accuracy']], 'accuracy', '
    Accuracy')
files = [COLAB_PATH_PREFIX + "neg.txt", COLAB_PATH_PREFIX + "
    pos.txt"]
for file in files:
    pred = model.predict(load_custom_text(load_file(file)))
    print(pred, "Positive" if pred > 0.5 else "Negative")
```