**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# ОТЧЕТ

**по лабораторной работе № 6**

**по дисциплине «Искусственные нейронные сети»**

**ТЕМА: Прогноз успеха фильмов по обзорам**

Студент гр. 8383 _____ Мололкин К.А.

Преподаватель _____ Жангиров 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)

**Выполнение работы**

В начале работы были загружены и векторизованы данные IMDB. Затем построена модель нейронной сети со следующей структурой:

```
model = Sequential()
model.add(layers.Dense(50, activation="relu", input_shape=(idim, )))
model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dense(1, activation="sigmoid"))
```

Рисунок 1 – модель

В табл. 1 представлены результаты исследования при различном размере вектора представления текста.

Таблица 1 - результаты исследования при различном размере вектора

| Размер | Точность | Потери |
|---|---|---|
| 1000 | 0.8594 | 0.3107 |
| 3000 | 0.8829 | 0.2807 |
| 7000 | 0.8925 | 0.2814 |
| 10000 | 0.8964 | 0.2614 |

После 3000 точность растет медленно, а скорость обучения падает значительно.

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: 0.9298098
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: 0.26321936

**Вывод**

В ходе выполнения лабораторной работы была обучена нейронная сеть, прогнозирующая успех фильмов по обзорам (Predict Sentiment From Movie Reviews). Так же была изучена передача текстовых данных в нейронную сеть.

# Приложение А

## Листинг программы

```
import numpy as np
from tensorflow.keras.models import Sequential
from keras import layers
from keras.datasets import imdb

idim = 10000

def vectorize(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results


def load_text(filename):
    data = []
    with open(filename, 'r') as file:
        for line in file.readlines():
            data += [w.strip(''.join(['.', ',', ':', ';', '!', '?', '(',
      ')'])).lower() for w in line.strip().split()]
    index = imdb.get_word_index()
    x_test = []
    for w in data:
        if w in index and index[w] < idim:
            x_test.append(index[w])
    x_test = vectorize([np.array(x_test)], idim)
    return x_test


(training_data, training_targets), (testing_data, testing_targets) =
      imdb.load_data(num_words=idim)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)

data = vectorize(data, idim)
targets = np.array(targets).astype("float32")

test_x = data[:10000]
test_y = targets[:10000]
train_x = data[10000:]
train_y = targets[10000:]

model = Sequential()
model.add(layers.Dense(50, activation="relu", input_shape=(idim, )))
model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dense(1, activation="sigmoid"))

model.compile(optimizer="adam", loss="binary_crossentropy",
      metrics=["accuracy"])
history = model.fit(train_x, train_y, epochs=2, batch_size=500,
      validation_data=(test_x, test_y))

print(np.mean(history.history["val_accuracy"]))
```

```
while True:
    print("Input filename with review or 0 to stop")
    filename = input()
    if filename != "0":
        prediction = model.predict(load_text(filename))
        print(f"{filename} - {prediction}")
    else:
        break
```