ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
**Московский институт электроники и математики**

# Моя первая нейросеть. Классификация электронных писем

Проектная работа студента образовательной программы бакалавриата
«Прикладная математика»
по направлению подготовки 01.03.04 ПРИКЛАДНАЯ МАТЕМАТИКА

Студент
Морозов Д.С.

Руководитель проекта
Попов Виктор Юрьевич

**Москва 2024г.**

# Содержание

# 1 АННОТАЦИЯ

Данная работа посвящена разработке нейросети для решения проблемы классификации электронных писем. Так как сейчас почта используется повсеместно, то данная проблема становится более и более актуальной

Для достижения цели, а именно разработки модели, были изучены методики создания и обучения нейронных сетей, в эксперементе я использовал датасет, в котором содержались тексты писем и их категории. Практическая значимость заключается в возможности использования данной модели для оптимизации взаимодействия с почтой.

# 2 ВВЕДЕНИЕ

В наше время большое количество людей повседневно используют электронную почту, она связывает между собой почти все сферы жизни, упрощает коммуникацию, а также обеспечивает надёжное хранение информации. Из-за этого очень важно грамотно организовывать ее использование. Каждый день пользователи отправляют друг другу миллионы сообщений, поэтому одним из способов повышения эффективности управления вашим почтовым ящиком является классификация писем. В этом проекте мы будем рассматривать процесс класификации писем с использованием техник искусственного интелекта.

Основвной целью проекта является и создание нейросетевой модели для классификации текстов, которая определяет к какой категории относится письмо. Для достижения цели нужно выполнить следующие задачи:

1. Изучение способов классификации текста
2. Поиск датасета на котором будет обучатся наша модель
3. Создание самой модели
4. Тестирование ее на выбранном датасете

Метод исследования заключается в использовании нейросетей, которые будут обучаться на некоторых данных и выявлять закономерности, которые помогут в классификации писем.

# 3 ОСНОВНАЯ ЧАСТЬ

## 3.1 Теория

Для начала необходимо поговорить про устройство нейронных сетей и разобраться в некоторых терминах. Нейронная сеть состоит из слоёв, которые состоят из нейронов. Слои бывают разные:

1. Входной слой - слой, который принимает исходные данные
2. Выходной слой - слой, который предоставляет результат работы с данными, например классификация
3. Скрытые слои, они находятся между входным и выходным слоем

У нейронов есть функции активации, они применяются выходным данным нейрона. Между нейронами есть связи - веса, процесс обучения нейронной сети состоит в правильном подборе этих весов. Сами нейросети бывают разных типов - свёрточные, рекурентные, полносвязные и многие другие.

Перед нами стоит задача классификации писем, для успешного её выполнения нам нужно работать с содержанием письма, то есть нам надо его обрабоботать. Но нейросеть умеет работать только с числами, а не с словами. Поэтому первоочередной задачей является представить тексты наших писем как набор чисел, для этого мы можем сначала разбить текст на фрагменты меньшей длины(обычно предложения, слова, символы), а затем представить их либо как числа, либо как векторы, такой процесс называется векторизацией.

Итак теперь давайте определимся на что мы будем анализировать текст. Как я уже писал выше, почта имеет большое значение в современном мире, многие пользователи часто сталкиваются с переполнением своих почтовых ящиков. Проблема состоит еще и в том, что иногда нам намеренно отправляют спам, который может ухудшить

нашу продуктивность. Так как это актуальная проблема я поставил задачу классификации писем с целью выявления там спама.

## 3.2 Первая нейронная сеть

Прежде чем заниматься написанием нейросети нужно найти данные, на которых мы будем ее обучать. Для нашей задачи я нашел датасет, который состоит из более чем 5000 электронных писем.

| Category | Massage |
|---|---|
| ham | Ok lar... Joking wif u oni... |
| spam | Customer service annoncement. You have a New Years delivery waiting for you.Please call 07046744435 now to arrange delivery |
| ... | ... |
| ham | U still going to the mall? |

Таблица 1: Датасет спам/ не спам

Файл csv в котором представлен наш датасет выглядит как показано в таблице 1, ham означает,что письмо хорошее, spam, что плохое. Для создания нейросети будем работать с библиотекой tensorflow. Для начала работы импортируем библиотеки:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

Первая библиотека необходима для работы с csv файлом, numpy для удобства работа с матрицами и другими математическими объектами, про остальные поговорим позже, они понадобятся для самой нейросети. Далее напишем следующий код, который считает данные.

```
data = pd.read_csv('/content/spam.csv')
data['Category'] = data['Category'].replace({'spam': 1, 'ham': 0})
```

Последняя строчка меняет значения spam и ham на 0 и 1, вот мы уже начали преобразовывать текст в числовые значения. Теперь давайте разделим данные на обучающий сет (80 %) и тренировочный (20%) с помощью команды:

```
train_data, test_data = train_test_split(data, test_size=0.2,
                                         random_state=8)
```

Сейчас нам нужно приступить к токенизации текста. Для этого мы используем Tokenizer из библиотеки tensorflow, в скобках можно прописать настройку разных параметров, однако я буду пользоватся их значениями по умолчанию, однако укажу максимальное количество слов 10 000

```
tokenizer = Tokenizer(num_words=10000)
```

Далее поработаем с нашим тестовым и проверочным наборами, извлечем данные из столбца Message и преобразуем их в списки, а далее на основе тренировочного списка создаем словарь, в котором у каждого слова есть индекс который обозначает его частоту употребления в тексте.

```
trainText = train_data['Message'].tolist()
testText = test_data['Message'].tolist()
tokenizer.fit_on_texts(trainText)
```

Давайте посмотрим какие слова у нас самые первые: ('i', 1) ('to', 2) ('you', 3) ('a', 4) ('the', 5)

Теперь с помощью функции pad_sequences приведем последовательности к одной и той же длине, в нашем случае 100 будет максимумом.

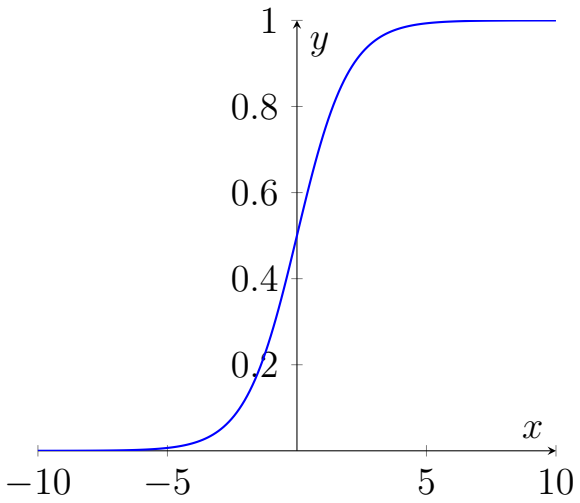
```
max_sequence_length = 100
train_sequences_padded = pad_sequences(train_sequences,
maxlen=max_sequence_length)

test_sequences_padded = pad_sequences(test_sequences,
maxlen=max_sequence_length)
```

Теперь можем заняться написанием архитектуры нашей сети. Я буду использовать полносвязную нейронную сеть, это такая где каждый нейрон данного слоя связан с каждым нейроном предыдущего.

```
model1 = Sequential()
model1.add(Dense(200, input_dim = 100 , activation="relu"))
model1.add(Dropout(0.25))
model1.add(BatchNormalization())
model1.add(Dense(1, activation='sigmoid'))
model1.compile(optimizer='adam',
               loss='binary_crossentropy', metrics=['accuracy'])
model1.summary()
```

Создаём последовательную модель, добавляем первый слой с 400 нейронами, на входе ожидаются векторы длины 100 и используем функцию активации relu, далее добавляем слой Dropout для борьбы с переобучением, далее слой BatchNormalization, он модифицирует выходные данные, способствует ускорению обучения. В конце добавляем выходной слой с 1 нейроном, так как у нас бинарная классификация.В конце

указываем оптимизатор adam, который обновляет веса модели на основе градиентного спуска, бинарная кросс-энтропия и метрика качества - доля правильных ответов. Функция 'sigmoid' принимает значения от 0 до 1, что как раз подходит для решения задачи бинарной классификации.Вот ее график.




А вот формула по которой рассчитывается значение бинарной кросс-энтропии:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)\right] \tag{1}$$

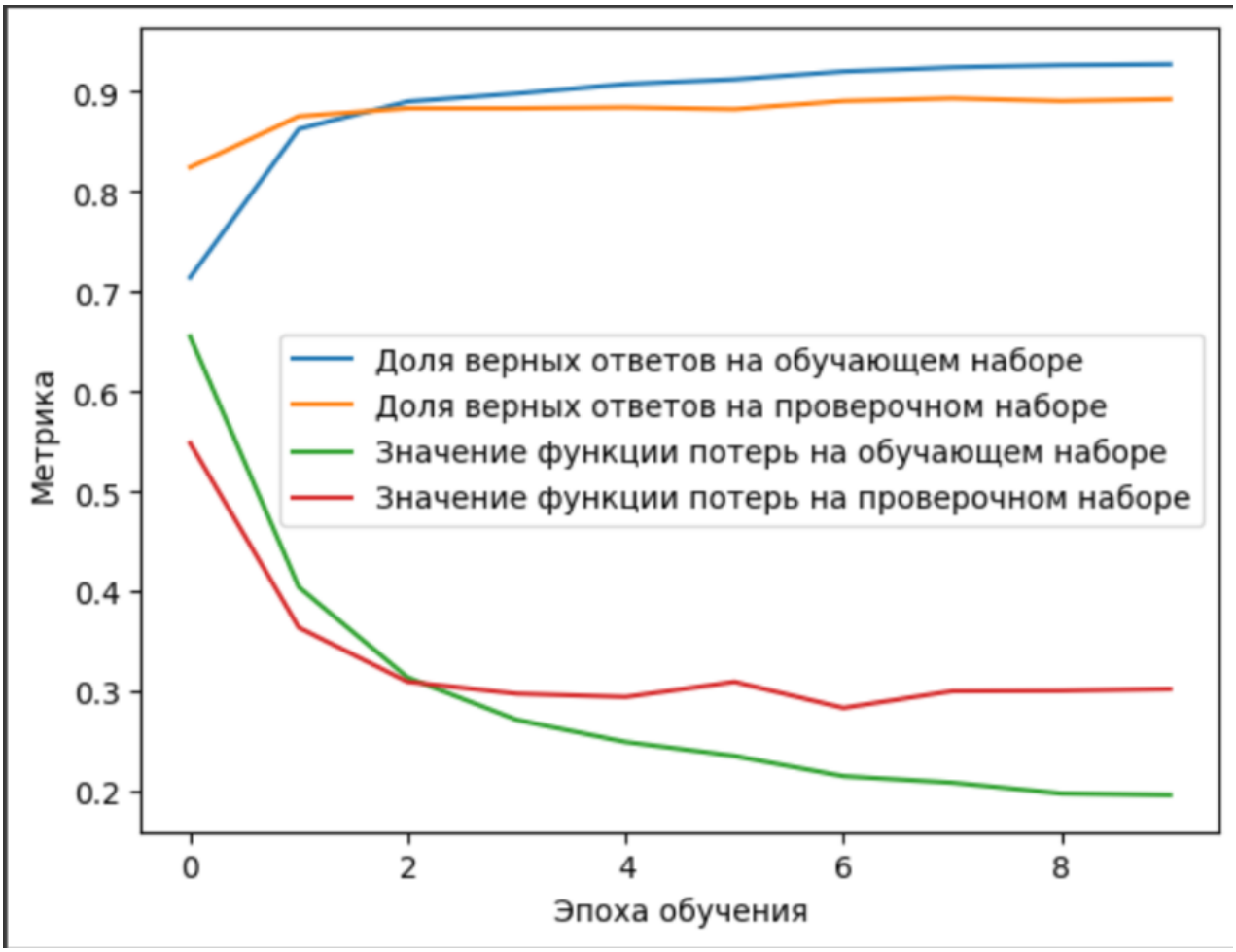
В формуле 1, $N$ - количество данных в нашем датасете, $y_i$ - истинное значение, $p_i$ - вероятность вычисленная нейросетью.

Теперь нам осталось приступить к обучению нашей модели. Напишем код:

```
import matplotlib.pyplot as plt


history = model1.fit(train_sequences_np,
                     train_data['Category'],
                     epochs=10,
                     batch_size=128,
                     validation_data=(test_sequences_np,
```

```
                    test_data['Category']))

plt.plot(history.history['accuracy'],
label='Доля верных ответов на обучающем наборе')
plt.plot(history.history['val_accuracy'],
label='Доля верных ответов на проверочном наборе')

plt.plot(history.history['loss'],
label='Значение функции потерь на обучающем наборе')
plt.plot(history.history['val_loss'],
label='Значение функции потерь на проверочном наборе')

plt.xlabel('Эпоха обучения')
plt.ylabel('Метрика')
plt.legend()
plt.show()
```

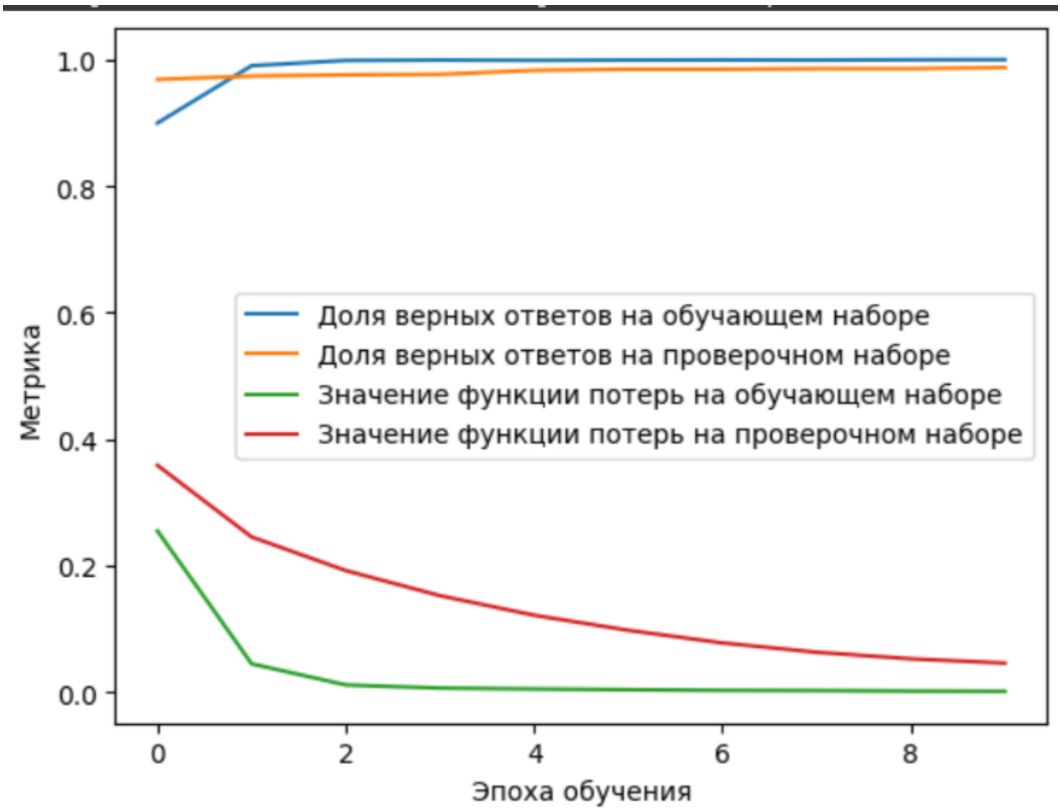
Количество эпох возьмём равным 10. Так же с помощью библиотеки matplotlib отобразим графики демонстриующие значение точности и функции потерь с течением эпох. Как мы видим точность достигает почти 90% а значение функции потерь уменьшается.




Однако можно доработать нашу нейросеть, посредством доведения данных до более приятного для нейросети вида. Тут мы довели наш текст до вида последовательности где вместо слов стоит их индекс в словаре, на практике же так редко делают, есть два метода которые между собой не сильно отличаются, первый - bag of words (bow), заключается он в том чтобы привести тексты в вектора у которых размерность равна числу слов в словаре. На позициях стоит 0 - если слова из словаря с таким же индексом в тексте нет, 1 - если есть. Второй способ называется embedding, всё сводится к представлению текста в виде вектора обычно с размерностью меньшей, чем в первом случае, засчёт того что тут могут быть не только 0 и 1, но другие числа. Давайте приведём наши данные к виду Bag of Words. Для этого напишем следущий код:

```
train_sequences_list = train_sequences_np.tolist()
xTrain_bow = tokenizer.sequences_to_matrix(train_sequences_list,
                                           mode='binary')
test_sequences_list = test_sequences_np.tolist()
xTest_bow = tokenizer.sequences_to_matrix(test_sequences_list,
                                          mode='binary')
```

Здесь мы переводим массивы numpy в списки, чтобы к ним применить операцию sequences_to_matrix, mode = 'binary' как раз и представляет наши последовательности в виде bag of words. Осталось внести правки в наши модель, теперь мы ожидаем на входе векторы длины 10 000. Позже я решил добавить еще одну метрику называемую полнотой, она показывает, Вот полученные результаты:



## 3.3 Вторая нейросеть

Вроде бы основная задача проекта выполнена, мы достигли хорошей точности, однако мне захотелось сделать еще что нибудь. Я начал думать, что же еще может быть полезным при работе с электроной почтой, облегчить взаимодействие с ней. Мне пришла такая идея, а что если еще и классифицировать письма по содержанию. Однако я не смог, как в предыдущем примере найти датасет. Решил сделать его сам. Для себя я выделил 4 основные категории: Работа, Личные, Уведомления от сервисов, Финансовые. Первая включает в себя письма о встречах, просьбах помочь по проектам и так далее, Вторая - письма от близки родственников, друзей. Третья - уведомления о подписках и выходах нового контента. Четвертая - уведомления от банков.Я создал датасет, состоящий из примерно 100 писем, так чтобы в каждой категории было примерно 24-26 писем, в этот раз он был на русском языке: Первые этапы с обработкой текста очень похожи на предыдущие, так же считываем данные меняем категории соответственно на 0,1,2,4, а затем преобразуем в наборы нулей и единиц

Так как здесь у нас очень маленький датасет, то я счёл интересным так же удалить еще и пару слов из текста по типу предлогов: в, на, о...

Создадим переменную в которую их запишем, так же в моём датасете используется часто слово привет,так что я решил его тоже удалить. Но для начала открыв файл я увидел, что кое-где были символы перевода на новую строку их тоже надо удалить.

```
prepositions = ['по', 'о', 'у', 'при', 'пре', 'до', 'на', 'Привет', 'в']
data_new['Сообщение']= data_new['Сообщение'].str.replace(r'\r|\n|\t','',
                                                    regex = True)
data_new['Сообщение'] = data_new['Сообщение'].apply(lambda x: ' '.join(
[word for word in x.split() if word.lower() not in prepositions]))
```

Далее так же создаем Tokenizer(), делим набор на тестовый и тренировочный, при-

| Категория | Сообщение |
|---|---|
| Работа | Мы завершили первый этап проекта "Построение сети связи нового поколения для обеспечения быстрой передачи данных"и хотели бы услышать ваше мнение и обратную связь. Пожалуйста поделитесь своими мыслями и предложениями по текущему состоянию проекта. |
| ... | ... |
| Финансовые | Уважаемый клиент мы вынуждены временно заблокировать ваш банковский счет из-за подозрительной активности. Пожалуйста обратитесь в нашу службу поддержки для получения дополнительной информации. |
| ... | ... |

Таблица 2: Датасет: Категории

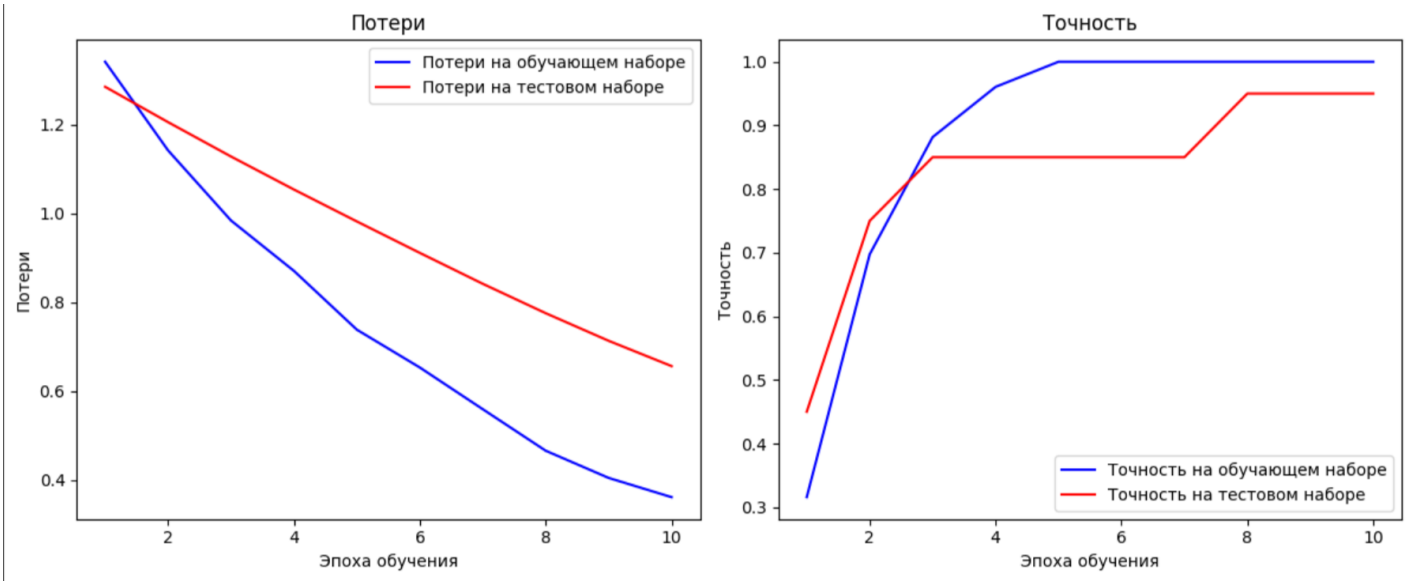
водим к типу Bag of Words, отличия будут только в архитектуре сети:

```
    model = Sequential()
model.add(Dense(100, input_shape=(1134,)))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(4))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy',
```

```
              optimizer='adam',
              metrics=['accuracy'])
```

```
print(model.summary())
```

Как видите выходной слой здесь имеет 4 нейрона, вместо одного, так же функция здесь другая, мы используем softmax. Давайте посмотрим на результаты:




Как мы видим точность достигает 95%, что конечно странно, потому что датасет у нас маленький, возможно я его не корректно составил, в любом случае вторая задача выполнена.

# 4 ЗАКЛЮЧЕНИЕ

Цель данного проекта выполнена была разработана модель способная классифицировать письма на наличие в них спама, а также модель, которая классиффицирует письма по 4 категориям. Обе модели демонстрируют довольно высокую точность при тестировании, однако вторая модель по моему мнению требует доработки и тестирования на других наборах данных, в работе также была дана небольшая теория про нейросети в целом.

# Список литературы


[1] Spam email classification — kaggle.com. https://www.kaggle.com/datasets/ashfakyeafi/spam-email-classification. [Accessed 19-05-2024].

[2] Классификация текстов с TensorFlow Hub: обзоры фильмов | TensorFlow Core — tensorflow.org. https://www.tensorflow.org/tutorials/keras/text$_c lassification_w ith_h ub?hl = ru.[Accessed 19 - 05 - 2024]$.

[3] Применение нейронных сетей в задачах обработки текстовых данных — cyberleninka.ru. https://cyberleninka.ru/article/n/primenenie-neyronnyh-setey-v-zadachah-obrabotki-tekstovyh-dannyh?ysclid=lwdzhhk6jx213379097. [Accessed 19-05-2024].

[4] Laxfed Paulacy. PYTHON—Text Classification with Keras in Python — medium.com. https://medium.com/paulacy-pulse/python-text-classification-with-keras-in-python-fde47414b985. [Accessed 19-05-2024].

[2] [1] [4] [3]