

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

Лабораторная работа № 6

тематика

Прогноз успеха фильмов по обзорам

Москва 2021

Цель

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

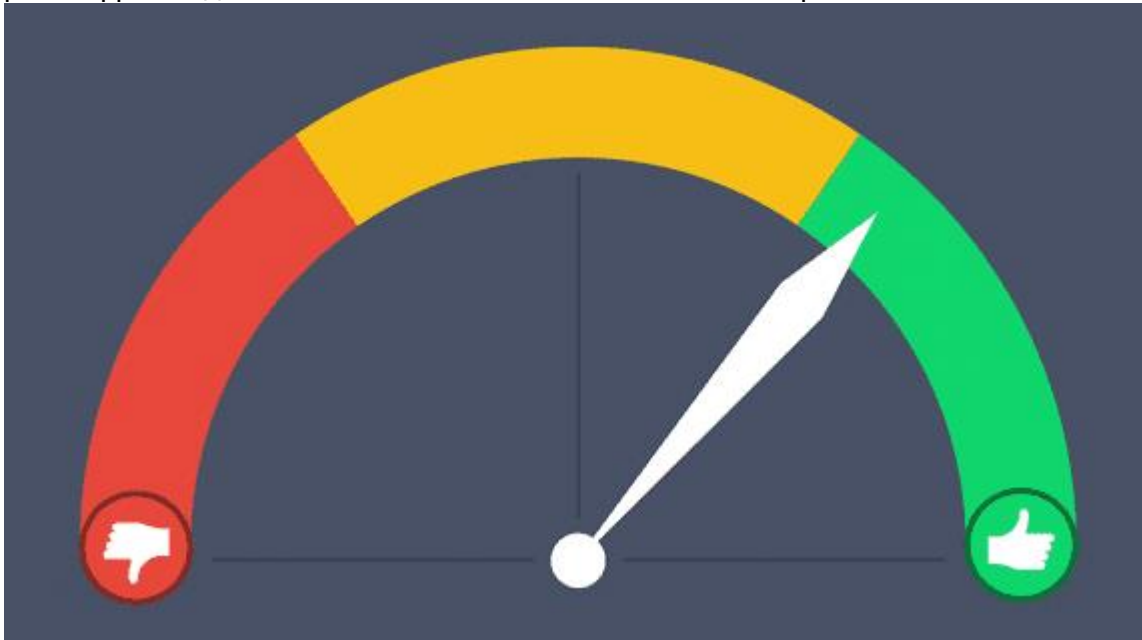
Задачи

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

Выполнение работы


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



During an adventure into the criminal underworld, Han Solo meets his future wife Chewbacca and encounters Lando Calrissian years before joining the Rebel Alliance.

Director: Ron Howard

Writers: Jonathan Kasdan, Lawrence Kasdan | [1 more credit »](#)

Stars: Alden Ehrenreich, Woody Harrelson, Emilia Clarke | [See full cast & crew »](#)

| | | |
|--|---|------------------------------|
| 62 Metascore From metacritic.com | Reviews 1,957 user 478 critic | Popularity 7 (↑ 1) |
|--|---|------------------------------|

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.

Импорт зависимостей и получение данных

Начнем с импорта необходимых зависимостей для предварительной обработки данных и построения модели.

```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from keras.utils import to_categorical
from keras import models
from keras import layers
```

Загрузим датасет IMDb, который уже встроен в Keras. Поскольку мы не хотим иметь данные обучения и тестирования в пропорции 50/50, мы сразу же объединим эти данные после загрузки для последующего разделения в пропорции 80/20:

```
from keras.datasets import imdb
(training_data, training_targets), (testing_data,
testing_targets) = imdb.load_data(num_words=10000)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets),
axis=0)
```

Изучение данных

Изучим наш датасет:

```
print("Categories:", np.unique(targets))
print("Number of unique words:",
len(np.unique(np.hstack(data))))
Categories: [0 1]
Number of unique words: 9998
length = [len(i) for i in data]
print("Average Review length:", np.mean(length))
print("Standard Deviation:", round(np.std(length)))
Average Review length: 234.75892
Standard Deviation: 173.0
```

Можно видеть, что все данные относятся к двум категориям: 0 или 1, что представляет собой настроение обзора. Весь датасет содержит 9998 уникальных слов, средний размер обзора составляет 234 слова со стандартным отклонением 173.

Рассмотрим простой способ обучения:

```
print("Label:", targets[0])
Label: 1
print(data[0])
```

```
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941,
4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 480,
284, 5, 150, 4, 172, 112, 167, 2, 336, 385, 39, 4, 172, 4536, 1111,
17, 546, 38, 13, 447, 4, 192, 50, 16, 6, 147, 2025, 19, 14, 22,
4, 1920, 4613, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15,
13, 1247, 4, 22, 17, 515, 17, 12, 16, 626, 18, 2, 5, 62, 386, 12,
8, 316, 8, 106, 5, 4, 2223, 5244, 16, 480, 66, 3785, 33, 4, 130,
12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 48, 25, 1415, 33, 6, 22,
12, 215, 28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 117, 5952,
15, 256, 4, 2, 7, 3766, 5, 723, 36, 71, 43, 530, 476, 26, 400,
317, 46, 7, 4, 2, 1029, 13, 104, 88, 4, 381, 15, 297, 98, 32, 2071,
56, 26, 141, 6, 194, 7486, 18, 4, 226, 22, 21, 134, 476, 26, 480,
5, 144, 30, 5535, 18, 51, 36, 28, 224, 92, 25, 104, 4, 226, 65,
16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16,
5345, 19, 178, 32]
```

Здесь вы видите первый обзор из датасета, который помечен как положительный (1). Нижеследующий код производит обратное преобразование индексов в слова, чтобы мы могли их прочесть. В нем каждое неизвестное слово заменяется на «#». Это делается с помощью функции `get_word_index()`.

```
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
data[0]] )
print(decoded)
```

```
# this film was just brilliant casting location scenery story
direction everyone's really suited the part they played and you
could just imagine being there robert # is an amazing actor and
now the same being director # father came from the same scottish
island as myself so i loved the fact there was a real connection
with this film the witty remarks throughout the film were great
it was just brilliant so much that i bought the film as soon as
it was released for # and would recommend it to everyone to watch
and the fly fishing was amazing really cried at the end it was so
sad and you know what they say if you cry at a film it must have
been good and this definitely was also # to the two little boy's
that played the # of norman and paul they were just brilliant
children are often left out of the # list i think because the stars
that play them all grown up are such a big profile for the whole
film but these children are amazing and should be praised for what
they have done don't you think the whole story was so lovely
because it was true and was someone's life after all that was
shared with us all
```

Подготовка данных

Пришло время подготовить данные. Нужно векторизовать каждый обзор и заполнить его нулями, чтобы вектор содержал ровно 10 000 чисел. Это означает, что каждый обзор, который короче 10 000 слов, мы заполняем нулями. Это делается потому, что самый большой обзор имеет почти такой же размер, а каждый элемент входных данных нашей нейронной сети должен иметь одинаковый размер. Также нужно выполнить преобразование переменных в тип `float`.

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

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

Разделим датасет на обучающий и тестировочный наборы. Обучающий набор будет состоять из 40 000 обзоров, тестировочный — из 10 000.

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

Создание и обучение модели

Теперь можно создать простую нейронную сеть. Начнем с определения типа модели, которую мы хотим создать. В Keras доступны два типа моделей: последовательные и с функциональным API.

Затем нужно добавить входные, скрытые и выходные слои. Для предотвращения переобучения будем использовать между ними исключение («dropout»). Обратите внимание, что вы всегда должны использовать коэффициент исключения в диапазоне от 20% до 50%. На каждом слое используется функция «dense» для полного соединения слоев друг с другом. В скрытых слоях будем использовать функцию активации «relu», потому что это практически всегда приводит к удовлетворительным результатам. Не бойтесь экспериментировать с другими функциями активации. На выходном слое используем сигмоидную функцию, которая выполняет перенормировку значений в диапазоне от 0 до 1. Обратите внимание, что мы устанавливаем размер входных элементов датасета равным 10 000, потому что наши обзоры имеют размер до 10 000 целых чисел. Входной слой принимает элементы с размером 10 000, а выдает — с размером 50.

Наконец, пусть Keras выведет краткое описание модели, которую мы только что создали.

```
# Input - Layer
model.add(layers.Dense(50, activation = "relu",
input_shape=(10000, )))
# Hidden - Layers
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"))
# Output- Layer
model.add(layers.Dense(1, activation = "sigmoid"))
model.summary()
model.summary()
```

| Layer (type) | Output Shape | Param # |
|---------------------|--------------|---------|
| dense_1 (Dense) | (None, 50) | 500050 |
| dropout_1 (Dropout) | (None, 50) | 0 |
| dense_2 (Dense) | (None, 50) | 2550 |
| dropout_2 (Dropout) | (None, 50) | 0 |
| dense_3 (Dense) | (None, 50) | 2550 |
| dense_4 (Dense) | (None, 1) | 51 |

Total params: 505,201
Trainable params: 505,201

Non-trainable params: 0

Теперь нужно скомпилировать нашу модель, то есть, по существу, настроить ее для обучения. Будем использовать оптимизатор «adam». Оптимизатор — это алгоритм, который изменяет веса и смещения во время обучения. В качестве функции потерь используем бинарную кросс-энтропию (так как мы работаем с бинарной классификацией), в качестве метрики оценки — точность.

```
model.compile(  
    optimizer = "adam",  
    loss = "binary_crossentropy",  
    metrics = ["accuracy"]  
)
```

Теперь можно обучить нашу модель. Мы будем делать это с размером партии 500 и только двумя эпохами, поскольку я выяснил, что модель начинает переобучаться, если тренировать ее дольше. Размер партии определяет количество элементов, которые будут распространяться по сети, а эпоха — это один проход всех элементов датасета. Обычно больший размер партии приводит к более быстрому обучению, но не всегда — к быстрой сходимости. Меньший размер партии обучает медленнее, но может быстрее сойтись. Выбор того или иного варианта определенно зависит от типа решаемой задачи, и лучше попробовать каждый из них. Если вы новичок в этом вопросе, я бы посоветовал вам сначала использовать размер партии 32, что является своего рода стандартом.

```
results = model.fit(  
    train_x, train_y,  
    epochs= 2,  
    batch_size = 500,  
    validation_data = (test_x, test_y)  
)
```

Train on 40000 samples, validate on 10000 samples

Epoch 1/2

40000/40000 [=====] - 5s 129us/step -
loss: 0.4051 - acc: 0.8212 - val_loss: 0.2635 - val_acc: 0.8945

Epoch 2/2

40000/40000 [=====] - 4s 90us/step -
loss: 0.2122 - acc: 0.9190 - val_loss: 0.2598 - val_acc: 0.8950

Проведем оценку работы модели:

```
print(np.mean(results.history["val_acc"]))  
0.894750000536
```

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)