

Отчет: Разработка веб-сервиса для оценки комментариев (отзывов) к фильмам.

План отчета

1. Вводная часть. Постановка задачи.
2. Загрузка данных.
3. Предобработка данных.
4. Разработка модели предсказания.
5. Разработка веб-сервиса.
6. Размещение веб-сервиса на сервере.
7. Итоги и предложения.

1. Вводная часть. Постановка задачи.

В рамках тестового задания предлагается разработать веб-сервис для оценки комментариев (отзывов) к фильмам. Мы предлагаем вам взять открытый набор данных, который содержит в себе отзывы о фильмах, а также соответствующие им оценки рейтинга. Рейтинг может служить ориентиром для построения модели классификации отзывов.

https://ai.stanford.edu/~amaas/data/sentiment/acllmbd_v1.tar.gz
(https://ai.stanford.edu/~amaas/data/sentiment/acllmbd_v1.tar.gz)

Для выполнения тестового задания вам необходимо:

1. Обучить модель на языке Python для классификации отзывов.
2. Разработать веб-сервис на базе фреймворка Django для ввода отзыва о фильме с автоматическим присвоением рейтинга (от 1 до 10) и статуса комментария (положительный или отрицательный).
3. Развернуть сервис в открытом доступе для оценки работоспособности прототипа.
4. Подготовить отчет о работе с оценкой точности полученного результата на тестовой выборке.
5. Отправить ответным письмом ссылку на прототип сервиса, ссылку на открытый репозиторий github с исходным кодом проекта, отчет о проделанной работе в формате pdf.

Предлагаю ознакомиться с видом завершеного веб-сервиса (п.5. Разработка веб-сервиса.):

← → ↻ Не защищено | sergeym.pythonanywhere.com

Предсказания рейтинга фильма по отзыву.

Получите предсказание рейтинга фильма по вашему отзыву в 10 бальной шкале, а также статус отзыва (плохой/хороший).

Введите свой отзыв на английском языке о фильме ниже:

Very good, i like it

Отправить

Мы вернем вам предсказание

← → ↻ Не защищено | sergeym.pythonanywhere.com/result/?csrfmiddlewaretoken=H

Предсказанный рейтинг

Рейтинг фильма по отзыву: 10 звёзд.

Статус отзыва: Положительный.

[Вернуться назад](#)

Использованные средства разработки: Google Colab, Jupyter notebook, PyCharm, бесплатный хостинг <https://www.pythonanywhere.com/> (<https://www.pythonanywhere.com/>).

2. Загрузка данных.

Для выполнения работы выбран набор данных Large Movie Review Dataset

https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz

(https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz) содержащий в себе 25.000

тренировочных, 25.000 тестовых отзывов о фильмах с IMDB, а также неразмеченные отзывы.

В качестве модели для работы мной был выбран тип нейросетевой модели. Установлены необходимые библиотеки и загружен датасет, Так как отзывы разделены на pos и neg, а метки рейтинга находятся в названии файлов, созданы новые папки с метками рейтинга и перераспределяем отзывы. Удалены неразмеченные отзывы и оставшиеся пустые папки. Стоит заметить что отзывы перераспределились на 8 папок - neg 1-4, pos 7-10, из-за отсутствия нейтральных отзывов (5,6). Это повлияло на мой выбор классификации на 8 классов (1-4, 7-10), а не 10 (1-10 звезд).

```
[ ] !ls aclImdb/train
```

```
1  2  4  8  labeledBow.feats  pos      urls_neg.txt  urls_unsup.txt
10 3  7  9  neg              unsupBow.feats  urls_pos.txt
```

```
[ ] for dir in range(1,5):
    os.mkdir(f'aclImdb/test/{dir}')
for dir in range(7,11):
    os.mkdir(f'aclImdb/test/{dir}')
!ls aclImdb/test
```

```
1 10 2 3 4 7 8 9 labeledBow.feats neg pos urls_neg.txt urls_pos.txt
```

Теперь расположим файлы в папках соответствующих их рейтингу и удалим папки pos, neg

```
▶ directory = 'aclImdb/train'

for root, dirs, files in os.walk(directory):
    if root == 'aclImdb/train/neg' or root == 'aclImdb/train/pos':
        for file_ in files:
            if int(file_[-5])!=0:
                dir_ = int(file_[-5])
            else:
                dir_ = int(file_[-6:-4])

            os.rename(''.join([root, '/', file_]), f'aclImdb/train/{dir_}/{file_}')
```

```
[ ] !rm -r aclImdb/train/neg
!rm -r aclImdb/train/pos
```

Построены датасеты для обучения модели используя `tf.keras.utils.text_dataset_from_directory`: `raw_train_ds` - 25.000 отзывов, `raw_ds` - 25.000 отзывов, `raw_test_ds` - 25.000 отзывов, метки преобразовались на числа 0-7.

```

batch_size = 32
raw_train_ds = tf.keras.utils.text_dataset_from_directory(
    "aclImdb/train",
    batch_size=batch_size,
    validation_split=0.2,
    subset="training",
    seed=1337,
)
raw_val_ds = tf.keras.utils.text_dataset_from_directory(
    "aclImdb/train",
    batch_size=batch_size,
    validation_split=0.2,
    subset="validation",
    seed=1337,
)
raw_test_ds = tf.keras.utils.text_dataset_from_directory(
    "aclImdb/test", batch_size=batch_size)

print(f"Number of batches in raw_train_ds: {raw_train_ds.cardinality()}")
print(f"Number of batches in raw_val_ds: {raw_val_ds.cardinality()}")
print(f"Number of batches in raw_test_ds: {raw_test_ds.cardinality()}")

```

```

Found 25000 files belonging to 8 classes.
Using 20000 files for training.
Found 25000 files belonging to 8 classes.
Using 5000 files for validation.
Found 25000 files belonging to 8 classes.
Number of batches in raw_train_ds: 625
Number of batches in raw_val_ds: 157
Number of batches in raw_test_ds: 782

```

3. Предобработка данных.

Перед обучением модели проведена отчистка и стандартизация отзывов. Для этого переведены все слова в нижний регистр, удалены стоп-слова, HTML теги, URL, возможные эмодзи, а также расшифрованы слова полученные слиянием двух слов (don't = do not).

```

[ ] # удалим стоп-слова
def remove_stopwords(text, label):
    text = tf.strings.regex_replace(text, r'\b(' + r'|'.join(stops) + r')\b\s*', "")
    return text, label

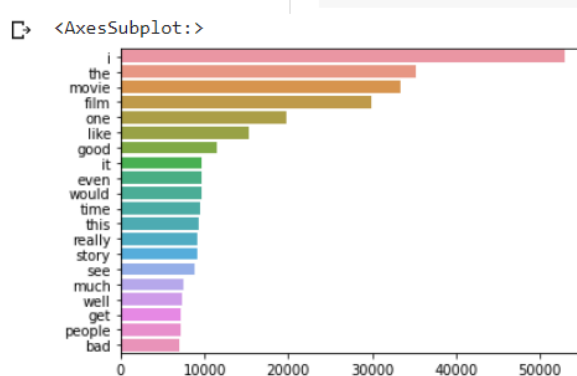
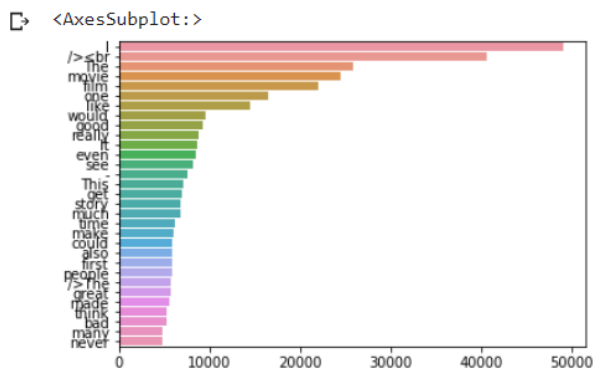
raw_train_ds, raw_val_ds, raw_test_ds = raw_train_ds.map(remove_stopwords), raw_val_ds.map(remove_stopwords), raw_test_ds.map(remove_stopwords)

[ ] # удалим теги
def remove_tags(text, label):
    return tf.strings.regex_replace(text, "<br />", " "), label

raw_train_ds, raw_val_ds, raw_test_ds = raw_train_ds.map(remove_tags), raw_val_ds.map(remove_tags), raw_test_ds.map(remove_tags)

```

На графиках ниже показана частота появления слов до и после предобработки.



Далее слова векторизированны для передачи в модель.

```
# Константы модели
max_features = 20000
embedding_dim = 128
sequence_length = 500

# слой векторизации
vectorize_layer = tf.keras.layers.TextVectorization(
    standardize=None,
    max_tokens=max_features,
    output_mode="int",
    output_sequence_length=sequence_length,
)

# адаптируем vectorize_layer к нашим отзывам
text_ds = raw_train_ds.map(lambda x, y: x)
vectorize_layer.adapt(text_ds)
```

4. Разработка модели предсказания.

Для обучения мной были выбраны 4 варианта нейросетей: 5-слойная сеть с 2 слоями Conv1D, 5-слойная сеть с 2 слоями LSTM, 4-слойная сеть с слоями Conv1D и Bidirectional LSTM, 3-слойная сеть с слоем Bidirectional LSTM.

```
# Conv1D модель
inputs = tf.keras.Input(shape=(None,), dtype="int64")
x = layers.Embedding(max_features, embedding_dim)(inputs)
x = layers.Dropout(0.5)(x)

x = layers.Conv1D(128, 7, padding="valid", activation="relu", strides=3)(x)
x = layers.Conv1D(128, 7, padding="valid", activation="relu", strides=3)(x)
x = layers.GlobalMaxPooling1D()(x)

x = layers.Dense(128, activation="relu")(x)
x = layers.Dropout(0.5)(x)
predictions = tf.keras.layers.Dense(8, name="predictions")(x)

model_conv = tf.keras.Model(inputs, predictions)
model_conv.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), optimizer="adam", metrics=["accuracy"])

# LSTM 2 слоя
inputs = tf.keras.Input(shape=(None,), dtype="int64")
x = layers.Embedding(max_features, embedding_dim)(inputs)

x = layers.LSTM(128, return_sequences=True, return_state=False)(x)
x = layers.LSTM(256, return_sequences=True, return_state=False)(x)
x = layers.GlobalMaxPooling1D()(x)
```

```
# Bidirectional LSTM
inputs = tf.keras.Input(shape=(None,), dtype="int64")
x = layers.Embedding(max_features, embedding_dim)(inputs)

x = layers.Conv1D(filters=32, kernel_size=3, padding='same', activation='relu')(x)
x = layers.AveragePooling1D(pool_size=2)(x)
x = layers.Bidirectional(LSTM(200, dropout=0.5))(x)

predictions = tf.keras.layers.Dense(8, name="predictions")(x)

model_bi = tf.keras.Model(inputs, predictions)
model_bi.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), optimizer="adam", metrics=["accuracy"])

# LSTM модель
inputs = tf.keras.Input(shape=(None,), dtype="int64")
x = layers.Embedding(max_features, embedding_dim)(inputs)

x = layers.Bidirectional(LSTM(75, dropout=0.1))(x)
predictions = tf.keras.layers.Dense(8, name="predictions")(x)

model_lstm = tf.keras.Model(inputs, predictions)
model_lstm.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), optimizer="adam", metrics=["accuracy"])
```

Во время обучения были выбраны 15 эпох обучения, что достаточно, так как все модели начали переобучаться.

На рисунке ниже показана оценка моделей на тестовой выборке:

```
[ ] model_conv.evaluate(test_ds)
```

```
782/782 [=====] - 6s 7ms/step - loss: 4.4273 - accuracy: 0.3440
[4.427250862121582, 0.3439599871635437]
```

```
[ ] model_LSTM2.evaluate(test_ds)
```

```
782/782 [=====] - 15s 19ms/step - loss: 6.1816 - accuracy: 0.3658
[6.181572437286377, 0.36583998799324036]
```

```
▶ model_bi.evaluate(test_ds)
```

```
↗ 782/782 [=====] - 9s 12ms/step - loss: 3.1960 - accuracy: 0.3191
[3.195984125137329, 0.31911998987197876]
```

```
[ ] model_LSTM.evaluate(test_ds)
```

```
782/782 [=====] - 12s 15ms/step - loss: 4.3684 - accuracy: 0.3100
[4.368371486663818, 0.3100000023841858]
```

Самая высокая точность оказалось у LSTM2 модели - 0.3658, но у нее и самая большая потеря 6.1816. При всем этом точность всех моделей оказывается сравнительно маленькой ≈ 0.35 , что гораздо выше случайного предсказания в 0.125!

Такая точность говорит о трудности классификации текста и необходимости доработки как модели, так и предобработки (7. Итоги и предложения.)

Все модели были преобразованы в end-to-end с добавлением слоя `vectorize_layer` для простоты развертывания и сохранены в SavedModel формате.

```
▶ #LSTM2 end-to-end
inputs = tf.keras.Input(shape=(1,), dtype="string")
indices = vectorize_layer(inputs)
outputs = model_LSTM2(indices)

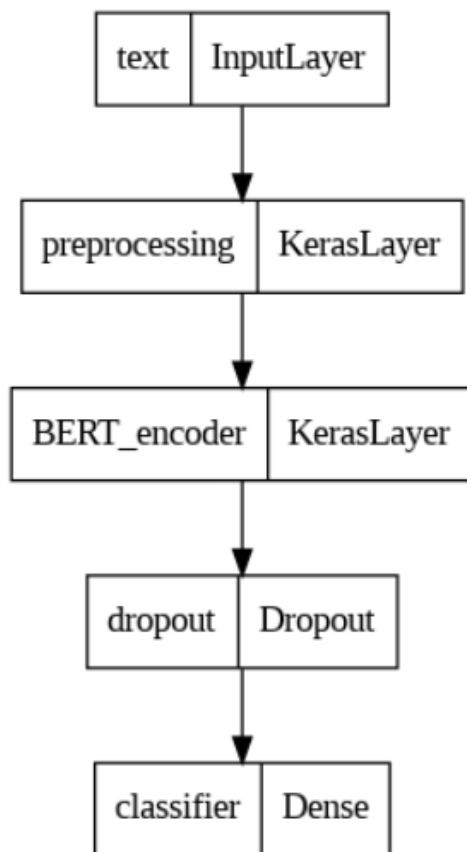
end_to_end_model = tf.keras.Model(inputs, outputs)
end_to_end_model.compile(
    loss='sparse_categorical_crossentropy', optimizer="adam", metrics=["accuracy"]
)

end_to_end_model.evaluate(raw_test_ds)
```

```
782/782 [=====] - 20s 24ms/step - loss: 6.5575 - accuracy: 0.3658
[6.557531833648682, 0.36583998799324036]
```

Для увеличения точности использована другая предобученная модель на основе трансформеров BERT `bert-base-uncased` с соответствующим ей препроцессором:

```
[ ] bert_model_name = 'talking-heads_base'
tfhub_handle_encoder = 'https://tfhub.dev/tensorflow/talkheads_ggelu_bert_en_base/1'
tfhub_handle_preprocess = 'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3'
```



```
[ ] loss, accuracy = bert_model.evaluate(raw_test_ds)
```

```
print(f'Loss: {loss}')
print(f'Accuracy: {accuracy}')
```

```
782/782 [=====] - 101s 128ms/step - loss: 1.6138 - accuracy:
Loss: 1.6138099431991577
Accuracy: 0.3978399932384491
```

Во время обучения модель также начала переобучаться, что снова говорит о необходимости доработки данных.

После 5 эпох дообучения модель улучшила точность предыдущих моделей на ≈ 0.05 до ≈ 0.4 .

5. Разработка веб-сервиса.

На основе полученной модели разработан веб-сервис на Django включающий в себя: 2 .html страницы с формой ввода и выводом рейтинга и статуса отзыва (положительный, отрицательный), сохраненную модель нейронной сети с 2LSTM слоями (так как весит меньше модели BERT), скрипты для передачи данных, загрузки модели, расчета.

← → ↻ ⚠ Не защищено | sergeym.pythonanywhere.com

Предсказания рейтинга фильма по отзыву.

Получите предсказание рейтинга фильма по вашему отзыву в 10 бальной шкале, а также статус отзыва (плохой/хороший).

Введите свой отзыв на английском языке о фильме ниже:

Very good, i like it

Отправить

Мы вернем вам предсказание

Активация Wi
Чтобы активировать
"Параметры".

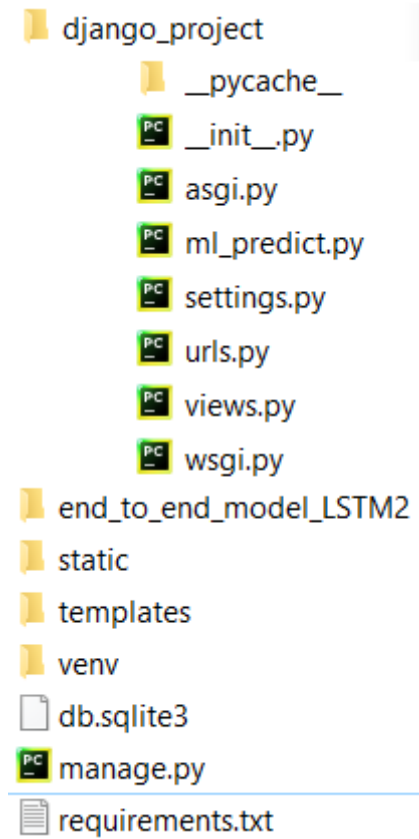
← → ↻ ⚠ Не защищено | sergeym.pythonanywhere.com/result/?csrfmiddlewaretoken=H

Предсказанный рейтинг

Рейтинг фильма по отзыву: 10 звёзд.

Статус отзыва: Положительный.


[Вернуться назад](#)



6. Размещение веб-сервиса на сервере.

Веб-сервис размещен на бесплатном хостинге <https://www.pythonanywhere.com/>
(<https://www.pythonanywhere.com/>)

Send feedback Forums Help Blog Pricing & sign up Log in

 **pythonanywhere**
by ANACONDA

Host, run, and code Python in the cloud!

Get started for free. Our basic plan gives you access to machines with a [full Python environment](#) already installed. You can develop and host your website or any other code directly from your browser without having to install software or manage your own server.

Need more power? Upgraded plans start at \$5/month.

Start running Python online in less than a minute! »

Watch our one-minute video »

Not convinced? [Read what our users are saying!](#)

pythonanywhere Dashboard Consoles

All done! Your web app is now set up. Details below.

example.pythonanywhere.com
www.mydomain.com
www.myotherdomain.com
Add a new web app

Configuration for [www.myotherdomain.com](#)
Reload:
Reload [www.myotherdomain.com](#)
DNS setup:
How to point your domain at your website.
CNAME: [webapp-4.pythonanywhere.com](#)
Traffic:
How busy is your site?
Hits per month
Hits per day

Для этого были переучены модели под tensorflow==2.9.0. установленный на хостинге, выделены все зависимости в requirements.txt и загружены на сервере, исправлены пути расположения файлов на сервере, также собран и запущен сам веб-сервис определения рейтинга отзыва
<http://sergeym.pythonanywhere.com/> (<http://sergeym.pythonanywhere.com/>)

pythonanywhere by ANACONDA

/home/ SergeyM

Directories

Enter new directory name [New directory](#)

- .cache/
- .keras/
- .local/
- .virtualenvs/
- django_ml/
- django_project/
- env/

Files

Enter new file name, eg hello.py [New file](#)

- .bashrc 2023-03-02 13:43 560 bytes
- .gitconfig 2023-03-02 13:43 266 bytes
- .profile 2023-03-02 13:43 79 bytes
- .pythonstartup.py 2023-03-02 13:43 77 bytes
- .vimrc 2023-03-02 13:43 4.6 KB
- README.txt 2023-03-02 13:43 232 bytes

[Upload a file](#)

100MiB maximum size

7. Итоги и предложения.

В соответствии с "п.1 Вводная часть. Постановка задачи" была проделана большая работа, задача обучения модели нейросети предсказания рейтинга и статуса отзыва о фильме и запуск веб-сервиса выполнены в полном объеме.

В качестве предложений для улучшения моделей с точностями 0.35 и 0.4 необходима доработка данных, а именно добавление нейтральных отзывов 5 и 6 звезд; использование n-грамм при предобработке; удаление не всех стоп-слов; использование другой векторизации например word2vec; лемматизация слов. Также хочу отметить в текущей задаче использовать регрессию вместо классификации, так как имеется числовое отображение отзыва от 1 до 10, а ассигасу не дает понимания близости предсказанного отзыва к правильному (например ошибка в одну, пол звезды), что можно видеть используя MSE/RMSE.

Спасибо за внимание!

E-mail: SergeyA.Misyura@yandex.ru (<mailto:SergeyA.Misyura@yandex.ru>) тел. +7(952)574-38-72 Git: <https://github.com/Sergey-Misyura> (<https://github.com/Sergey-Misyura>)