Содержание

Проект для «Викишоп»

Интернет-магазин «Викишоп» запускает новый сервис. Теперь пользователи могут редактировать и дополнять описания товаров, как в вики-сообществах. То есть клиенты предлагают свои правки и комментируют изменения других. Магазину нужен инструмент, который будет искать токсичные комментарии и отправлять их на модерацию.

Обучите модель классифицировать комментарии на позитивные и негативные. В вашем распоряжении набор данных с разметкой о токсичности правок.

Постройте модель со значением метрики качества *F1* не меньше 0.75.

Инструкция по выполнению проекта

1. Загрузите и подготовьте данные.
2. Обучите разные модели.
3. Сделайте выводы.

Для выполнения проекта применять *BERT* необязательно, но вы можете попробовать.

Описание данных

Данные находятся в файле `toxic_comments.csv` . Столбец *text* в нём содержит текст комментария, а *toxic* — целевой признак.

Подготовка

In [1]:

```
!pip install catboost
```

```
Requirement already satisfied: catboost in c:\users\saidd\anaconda3\lib\site
-packages (0.25.1)
Requirement already satisfied: pandas>=0.24.0 in c:\users\saidd\anaconda3\li
b\site-packages (from catboost) (1.0.5)
Requirement already satisfied: six in c:\users\saidd\anaconda3\lib\site-pack
ages (from catboost) (1.15.0)
Requirement already satisfied: graphviz in c:\users\saidd\anaconda3\lib\site
-packages (from catboost) (0.16)
Requirement already satisfied: numpy>=1.16.0 in c:\users\saidd\anaconda3\lib
\site-packages (from catboost) (1.18.5)
Requirement already satisfied: scipy in c:\users\saidd\anaconda3\lib\site-pa
ckages (from catboost) (1.5.0)
Requirement already satisfied: plotly in c:\users\saidd\anaconda3\lib\site-p
ackages (from catboost) (4.9.0)
Requirement already satisfied: matplotlib in c:\users\saidd\anaconda3\lib\si
te-packages (from catboost) (3.2.2)
Requirement already satisfied: pytz>=2017.2 in c:\users\saidd\anaconda3\lib
\site-packages (from pandas>=0.24.0->catboost) (2020.1)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\users\saidd\anac
onda3\lib\site-packages (from pandas>=0.24.0->catboost) (2.8.1)
Requirement already satisfied: retrying>=1.3.3 in c:\users\saidd\anaconda3\l
ib\site-packages (from plotly->catboost) (1.3.3)
Requirement already satisfied: cycler>=0.10 in c:\users\saidd\anaconda3\lib
\site-packages (from matplotlib->catboost) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\saidd\anaconda3
\lib\site-packages (from matplotlib->catboost) (1.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\users\saidd\anaconda3\lib\site-packages (from matplotlib->catboost) (2.4.
7)
```

In [2]:

```
#%pip install ipykernel
```

In [3]:

```
# импортну все что свет видал, а там посоттрим что нужно а что нет)
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np
from string import punctuation
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.snowball import SnowballStemmer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import f1_score
```

In [4]:

```
data = pd.read_csv('toxic_comments.csv', error_bad_lines=False, engine="python")
data.head()
```

Out[4]:

	text	toxic
0	Explanation\nWhy the edits made under my usern...	0
1	D'aww! He matches this background colour I'm s...	0
2	Hey man, I'm really not trying to edit war. It...	0
3	"\nMore\nI can't make any real suggestions on ...	0
4	You, sir, are my hero. Any chance you remember...	0

In [5]:

```
try:
    data = pd.read_csv('toxic_comments.csv', error_bad_lines=False, engine="python")

except:
    data = pd.read_csv('/datasets/toxic_comments.csv', error_bad_lines=False, engine="pytho
data.head()
```

Out[5]:

	text	toxic
0	Explanation\nWhy the edits made under my usern...	0
1	D'aww! He matches this background colour I'm s...	0
2	Hey man, I'm really not trying to edit war. It...	0
3	"\nMore\nI can't make any real suggestions on ...	0
4	You, sir, are my hero. Any chance you remember...	0

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   text    159571 non-null  object
 1   toxic   159571 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 2.4+ MB
```

In [7]:

```
data.columns
```

Out[7]:

```
Index(['text', 'toxic'], dtype='object')
```

Данные прочитаны, инфа выведена

In [8]:

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\saidd\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\saidd\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[8]:

```
True
```

In [9]:

```
import string
punctuation = string.punctuation
```

In [10]:

```
noise = stopwords.words('english') + list(punctuation) + list('1234567890')
def tokenizer(value):
    noise = stopwords.words('english') + list(punctuation) + list('1234567890')
    value = value.lower()
    a = word_tokenize(value)
    b = list()
    for el in a:
        if el not in noise:
            if el.isdigit() == False:
                if not el[0].isdigit():
                    b.append(el)
    stemmer = SnowballStemmer('english')
    stemmed_example = [stemmer.stem(w) for w in b]
    a = ' '.join(stemmed_example)
    return a
```

In [11]:

```
%%time
data['text'] = data['text'].apply(tokenizer)
data.head()
```

```
Wall time: 9min 35s
```

Out[11]:

	text	toxic
0	explan edit made usernam hardcor metallica fan...	0
1	d'aww match background colour 'm seem stuck th...	0
2	hey man 'm realli tri edit war 's guy constant...	0
3	`` ca n't make real suggest improv wonder sect...	0
4	sir hero chanc rememb page 's	0

Подготовили данные для векторизации, использовав стемминг. Также удалили ненужные слова и символы

P.S.Ячейка кода выше чет долго стала выполняться, может занять до 10 минут

Обучение

CountVectorizer Edition

Делим выборку на обучающуюся и валидационную для выбора наилучшей модели

In [12]:

```
data_cv = data.copy()
x_train, x_val, y_train, y_val = train_test_split(data_cv.drop('toxic', axis=1),
                                                  data_cv['toxic'], test_size=0.35, random_
```

Векторизируем данные, используя N грамму с размерностью от 1 до 2. Так у нас будут различные комбинации и больше данные для моделей

In [13]:

```
count_vec = CountVectorizer(ngram_range=(1,2))
x_train_count = count_vec.fit_transform(x_train['text'])
x_train_count
```

Out[13]:

```
<103721x1760255 sparse matrix of type '<class 'numpy.int64'>'
        with 6061886 stored elements in Compressed Sparse Row format>
```

In [14]:

```
print(x_train_count.shape)
print(x_train.shape)
```

```
(103721, 1760255)
(103721, 1)
```

Трансформируем валидационную выборку

In [15]:

```
x_val_count = count_vec.transform(x_val['text'])
x_val_count.shape
```

Out[15]:

```
(55850, 1760255)
```

Используем модель SVM SGD.

In [16]:

```
model_sgd1 = SGDClassifier(class_weight='balanced', random_state=131, loss='hinge')
model_sgd1.fit(x_train_count, y_train)
pred1 = model_sgd1.predict(x_val_count)
f1_score(y_val, pred1)
```

Out[16]:

```
0.7818863879957128
```

Она справилась с поставленной задачей и дала нужный результат. Рассмотрим также модель Логистической регрессии

In [17]:

```
model_sgd2 = SGDClassifier(class_weight='balanced', random_state=131, loss='log')
model_sgd2.fit(x_train_count, y_train)
pred2 = model_sgd2.predict(x_val_count)
f1_score(y_val, pred2)
```

Out[17]:

```
0.7741500042183415
```

На валидационной выборке результат хорош, но SVM смотрелся лучше

Проверим SVM SGD на **тестовой выборке**

In [18]:

```
x_train, x_test, y_train, y_test = train_test_split(data_cv.drop('toxic', axis=1),
                                                    data_cv['toxic'], test_size=0.3, random_s
```

In [20]:

```
x_train_count = count_vec.fit_transform(x_train['text'])
x_test_count = count_vec.transform(x_test['text'])
```

In [21]:

```
#model_sgd1 = SGDClassifier(class_weight='balanced', random_state=131, loss='hinge', n_jobs
#model_sgd1.fit(x_train, y_train)
model_sgd1.fit(x_train_count, y_train)
predt = model_sgd1.predict(x_test_count)
print('F1 Score on test data:',f1_score(y_test, predt))
```

```
F1 Score on test data: 0.7887890005288207
```

Выводы

Мы добились требуемой точности, с помощью SGD Classifier hinge loss. До этого мы рассматривали Countvectorizer с N граммой равной (1,1), но как только мы добавили двойные сочетаний, то сразу качество увеличилось. А Логистическая регрессия показала себя немного хуже, нежели SVM

Чек-лист проверки

- ☑ Jupyter Notebook открыт
- ☑ Весь код выполняется без ошибок
- ☑ Ячейки с кодом расположены в порядке исполнения
- ☑ Данные загружены и подготовлены
- ☑ Модели обучены
- ☑ Значение метрики *F1* не меньше 0.75
- ☑ Выводы написаны

In [ ]:

```

```