Отчет по проекту Data Science: оценка пола автора отзыва о лекарстве по тексту отзыва.

Выполнили: студенты группы 11-402
Ринат Ханов и Дмитрий Евдокименко.

2017

Содержание

Введение

В каждый момент времени пользователями сети Интернет создается новая информация в таких объемах, что ее оценка экспертами и анализ “вручную” потребует излишние расходы и ресурсы. Именно по этой причине задачу оценки и анализа представляется выгодным автоматизировать с помощью методов машинного обучения.

Одно из применений текстового анализа — это оценка отзывов пользователей о тех или иных продуктах, например, лекарственных препаратах. Данный инструмент может быть использован врачами и фармацевтами для представления общей картины об эффективности какого-либо препарата или лекарства.

Была выбрана следующая задача — определение конкретных характеристик автора обзора, таких как пол, по тексту обзора о лекарстве. В этом отчете изложено подробнее о проделанной работе по проекту и полученных результатах.

Основные этапы работы

Для реализации поставленной задачи было выбрано использовать стандартные методы машинного обучения, поскольку именно они зачастую показывают наилучшие результаты в подобных задачах текстового анализа.

Рассмотрим основные этапы работы.

Извлечение информации

Первостепенная задача перед началом процесса машинного обучения — подготовка входных данных, а именно извлечение информации. В нашем случае, это извлечение текстов обзоров и информации об авторе с форума askapatient.com.

Для этого был реализован следующий алгоритм:

1. Загрузка всех страниц вида http://www.askapatient.com/drugalpha.asp?letter=A для каждой буквы английского алфавита. (страницы с ссылками на лекарства)
2. Обработка таблицы и извлечение ссылок на лекарства, начинающихся на данную букву.
3. Для каждого лекарства, загрузка страницы отзывов и сохранение их в файл формата CSV.

На данном этапе работы техническую сложность вызвало автоматическое определение поисковых роботов, реализованное на данном форуме. При попытке сделать множество веб-запросов к форуму с одного устройства, система определяет подозрительную активность и блокирует веб-запросы на определенное время. Обойти данное ограничение представилось возможным с помощью использования списка IP прокси-серверов в случайном порядке и сервиса Google Web Cache (загрузка страниц вида 'http://webcache.googleusercontent.com/search?q=cache:http://www.askapatient.com/drugalpha.asp?letter=A').

Результатом работы на данном этапе стал набор CSV файлов с отзывами по каждому лекарству. Общее количество лекарств — 1863, общее количество обзоров — более 150 тысяч.

Подготовка входных данных

Перед тем как приступить к обучению, нужно привести данные к определенному виду. Существует несколько способов представления данных для обучения модели. В нашей работе мы использовали несколько способов, чтобы затем, сравнивая результаты, выбрать наилучший.

Список использованных способов:

- Bag of words (вектор, содержащий количество упоминаний каждого слова в тексте отзыва)
- Words + bigrams (слова и пары слов)
- Bigrams (пары слов)
- Character n-grams (выборки по n идущих подряд символов из текста)

В конечном итоге, по результатам экспериментов в качестве набора признаков был выбран способ “Words + bigrams” как наиболее точный и стабильный.

Для начала из исходного набора отзывов были исключены отзывы о лекарствах, целевой аудиторией которых является только один конкретный пол, а также отзывы содержащие менее десяти слов, так как вероятность верного определения пола по таким коротким отзывам крайне мала.

Затем, для того чтобы отсеять нейтральные слова и пары слов, не относящиеся к конкретному полу, был применен алгоритм TF-IDF, дописывающий каждому слову или паре слов вес, определяющий степень влияния этого слова на пол автора текста.

По получившемуся набору слов и биграмм был сформирован набор данных, на которых производилось обучение моделей. В процессе обучения различных моделей, данный набор данных корректировался для улучшения эффективности обучения. Например, менялся размер набора признаков (минимальный вес tf-idf, необходимый для попадания признака в обучающий корпус), минимальное количество слов в отзыве необходимое для попадания в обучающий корпус, также корректировался набор лекарств, отзывы которых используются.

В итоге были выбраны следующие параметры обучающей выборки:

- Минимальное число слов в отзыве - 10
- Длина набора признаков - 10 000 (при большом числе признаков возникала проблема переобучения)
- Количество лекарств в наборе - 10% наиболее популярных лекарств.

Обучение модели

Для обучения модели были использованы методы сторонней библиотеки **scikit-learn**, реализованные на языке программирования Python.

Была проведена серия экспериментальных исследований с выбором различных классификаторов, таких как логистическая регрессия, метод опорных векторов, градиентный спуск, наивный байесовский классификатор и другие, а также различных комбинаций параметров для каждого вида классификатора.

Полученные результаты выявили наиболее точный и стабильный классификатор для входных данных этой задачи — логистическая регрессия со следующими параметрами:

- Точность, при которой обучение останавливается - **0.0001**
- Максимальное число итераций, после которых обучение останавливается - **1000**

В большинстве случаев обучение останавливается при достижении заданной точности, поэтому дальнейшее увеличение числа итераций не целесообразно. При увеличении точности, процесс обучения значительно замедляется, но при этом не дает значительного прироста эффективности классификатора (~ 0.001%).

Результаты

Для определения эффективности обученной модели была использована перекрестная проверка. Это метод оценки аналитической модели и её поведения на независимых данных. При оценке модели имеющиеся в наличии данные разбиваются на k частей. Затем на k−1 частях данных производится обучение модели, а оставшаяся часть данных используется для тестирования.

Средняя точность (accuracy) обученной модели при использовании перекрестной проверки — 0.75.

Результаты применения обученной модели с использованием логистической регрессии.

Пол	Precision	Recall
Женский	0.74	0.77
Мужской	0.77	0.74

Пояснение используемой терминологии:

Accuracy — отношение количества правильно классифицированных отзывов к общему количеству отзывов.

Precision для мужского пола — отношение количества фактически мужских отзывов, классифицированных алгоритмом как мужские, к общему количеству отзывов, выбранных алгоритмом как мужские.

Recall для мужского пола — отношение количества фактически мужских отзывов, классифицированных алгоритмом как мужские, к общему количеству фактически мужских отзывов в наборе данных.

Аналогично для женских отзывов.

Заключение

Как показывают результаты проделанной работы, задача анализа текстов и оценки характеристик их авторов, в частности пола автора, может быть достаточно эффективно решена с помощью методов машинного обучения.

Теоретически точность алгоритма в будущем можно улучшить с помощью использования других подходов машинного обучения, таких как нейронные сети или классификаторы, настроенные другими метапараметрами.

При анализе полученной точности следует также обратить внимание на то, что некоторая часть текстов написана в нейтральном стиле, и даже эксперт может допустить ошибку при определении пола автора такого отзыва.

Полученная точность алгоритма не является идеальной, как и практически в любых задачах по машинному обучению, однако она является достаточно высокой для получения приближенных результатов с минимальной затратой на человеческие ресурсы при необходимости обработать большое количество текстов отзывов.

Список использованных источников

- https://www.coursera.org/learn/machine-learning
- https://habrahabr.ru/post/149605/
- https://habrahabr.ru/company/preply/blog/216729/
- https://en.wikipedia.org/wiki/Tf–idf
- https://en.wikipedia.org/wiki/Machine_learning

Приложение

1. Исходный код блока в котором производится подготовка данных для обучения.
2. Исходный код блока в котором производится обучение модели и анализ полученных результатов.

1.

```python
import pandas as pd
import os, re

def parse_csv_dir_to_drugs_stat(dir): # collecting drugs statistics
    data = pd.DataFrame(columns=['Name', 'Reviews count', 'F', 'M'])
    ind = 0

    def process_drug(drug_data):  # one drug statistics
        drug_data.fillna("", inplace=True)
        ln = len(drug_data) # reviews count
        female_count = len(drug_data[drug_data["Sex"] == "F"]) #female reviews
        return ln, female_count, ln - female_count

    for i in os.listdir(dir): #for each drug in dir collecting all drugs
        print(i)
        drug = pd.DataFrame.from_csv(dir + i) #collecting one drug reviews
        data.loc[ind] = ([i] + list(process_drug(drug)))
        ind += 1
    data.sort_values('Reviews count', inplace=True, ascending=False) # sort by reviews count
    return data

def choose_drugs(drug_data):  # Drug processing. remove one gendered drug. get the top
    top_percent = 1.0
    return drug_data['Name'][:round(top_percent * len(drug_data))]  # top 10% drugs

def parse_drugs():  # collecting drugs
    dir = "out/drugs/"
    drugs = pd.DataFrame.from_csv("drugs_stat.csv")
    data = pd.DataFrame()
    print("parsing drugs dir...")
    for i in choose_drugs(drugs):
        new_data = pd.DataFrame.from_csv(dir + i)
        data = data.append(new_data)
    return data

def prepare_data(): # processing raw data to Dataset
    data = parse_drugs()
    data = data.dropna()

    def len_check(string):  #condition for min review length
        return len(string.split()) > 20

    data['Review'] = data['Side Effects'] + " " + data['Comments'] # review = comments + side effects text
    data['Review'] = data['Review'].apply(str.lower)

    criteria = data['Review'].apply(len_check) # adding len check to criteria

    print(len(data))
    data = data[criteria] # applying criteria
    print(len(data))

    men = data[data['Sex'] == 'M']
    print(len(men))
    women = data[data['Sex'] == 'F'].sample(len(men))

    data = men.append(women).sample(frac=1)  # reshuffle reviews, just in case.

    # regex = re.compile("([^a-zA-Z']|_)")

    def numify_sex(gender): # proceesing letter based sex to number based
        return 0 if gender == "F" else 1

    data['Sex'] = data['Sex'].apply(numify_sex)

    def replace_non_letters(word): # replacing non letter characters with spaces
        return re.sub("([^a-zA-Z']|_)", " ", word)

    data['Review'] = data['Review'].apply(replace_non_letters)
    return data

dataset = prepare_data()[['Sex', 'Review']]
dataset.to_csv("corpus.csv")
```

2.

```
from sklearn.cross_validation import KFold
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
import pandas as pd
from sklearn.metrics import classification_report
from sklearn import linear_model, cross_validation, svm
from sklearn import metrics
from sklearn import cross_validation
from sklearn import linear_model, preprocessing
import os, re, functools
import numpy as np

dataset = pd.DataFrame.from_csv("corpus.csv") # collecting dataset from csv
print(len(dataset))

clf = Pipeline([('vect',  # processing raw review text into feature set
        TfidfVectorizer(ngram_range=(1, 2), # words and bigrams
        max_features=None)) # no features number limit

    # , ('clf', linear_model.SGDClassifier(penalty='l2', n_jobs=-1, alpha=0.0003))
    #    , ('clf', svm.SVC())

            , ('clf', #Logistic regression classifier
            linear_model.LogisticRegression(n_jobs=-1)) # multitasking enable

                # , ('clf', MultinomialNB())

                ])

# X y split
X = dataset['Review']
y = dataset['Sex']

print("training...")


# Train/test split
X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.1)
clf.fit(X_train, y_train) # learining process (fitting vectorizer and classfier)


# cross-validation by Rinat
# scores = cross_validation.cross_val_score(clf, X, y, cv=5)

# K-fold CV
# k_fold = KFold(n=len(X), n_folds=10)
# scores = []
# for train_indices, test_indices in k_fold:
#     local_train_X = X.iloc[train_indices]
#     local_train_y = y.iloc[train_indices]
#
#     local_test_X = X.iloc[test_indices]
#     local_test_y = y.iloc[test_indices]
#
#     clf.fit(local_train_X, local_train_y)
#     predictions = clf.predict(local_test_X)
#
#     score = metrics.accuracy_score(local_test_y, predictions)
#     scores.append(score)
#     print(score)
# #
# scores = np.array(scores)
#
print(len(clf.named_steps['vect'].get_feature_names()))

y_pred = clf.predict(X_test) # predicting on test dataset

# creating accuracy +precision/recall report
report = metrics.classification_report(y_test, y_pred, target_names=["F", "M"])
print(report)

# print("CV accuracy: %0.3f (+/- %0.3f)" % (scores.mean(), scores.std() * 2))
print("Full corpus accuracy : %0.3f" % metrics.accuracy_score(y_test, y_pred))
```