Правительство Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Отчёт к курсовой работе

по дисциплине

«Организационное и правовое обеспечение информационной безопасности»

Работу выполнил

Студент группы СКБ222 _____ П. А. Тюняткин
подпись, дата

Работу проверил _____ А. Б. Лось
подпись, дата

Тема работы: «Фишинг – актуальная угроза личности и организации в целом – цели, способы осуществления, классификация по уровням угроз – шкала Фиша, методы противодействия.»

Москва 2024 г.

Оглавление

Введение

Рассмотрим основной и самый наглядный метод классификации текстов на СПАМ / НЕ СПАМ методами машинного обучения: векторизация текстов методом «Мешок слов», бинарная классификация их логистической регрессией.

Основная часть

Векторизация текстов методом «Мешок слов»

В данном методе создается пронумерованный словарь уникальных слов из датасета, либо словарь из n слов, но слова, не попавшие в словарь, не будут обработаны. Каждое слово представляется в виде вектора, размерность которого равна размерности словаря. Векторное представление слова таким методом – разреженный вектор, где все элементы равны 0, кроме i-ого элемента, который соответствует этому слову в словаре:

Словарь:

1. Арбуз
2. Ананас
3. …

i. Машина

…

Арбуз = [1, 0, 0, ..., 0]
Ананас = [0, 1, 0, ..., 0]
Машина = [0, 0, 0, ..., 1, ..., 0]

i-ая координата

Вектор предложения – сумма векторов его слов.

Пример:

Словарь:

1. Привет
2. И
3. Ещё
4. Арбуз
5. Машина
6. Ананас
7. Раз

1. Привет и ещё раз привет

«Мешок слов» для этого предложения:

1. [2, 1, 1, 0, 0, 0, 1]
   ↑ ↑ ↑ ↑
   1 2 3 7

Бинарная логистическая регрессия

Бинарная логистическая регрессия – это статистический метод, используемый для моделирования зависимости между зависимой переменной, которая принимает бинарные значения из конечного множества, такие как 0 и 1, и несколькими независимыми переменными (в нашем случае количество независимых переменных равно количеству слов в словаре).

Введем обозначения: n – количество объектов в выборке, m – количество признаков объекта, X (n x m) – матрица признаков объектов, W (m x 1) – матрица весов, Y (n x 1), ($y_i \in \{0, 1\}$) – матрица целевых переменных.

Тогда модель логистической регрессии для бинарной классификации принимает вид:

$$P(y_i = 1|x_i, W) = \sigma(< x_i, W >) = \frac{1}{1 + e^{-<x_i, W>}}$$

$$P(y_i = 0|x_i, W) = 1 - \sigma(< x_i, W >) = 1 - \frac{1}{1 + e^{-<x_i, W>}}$$

$$P(y_i|x_i, W) = (\frac{1}{1 + e^{-<x_i, W>}})^{y_i} \times (1 - \frac{1}{1 + e^{-<x_i, W>}})^{1-y_i}$$

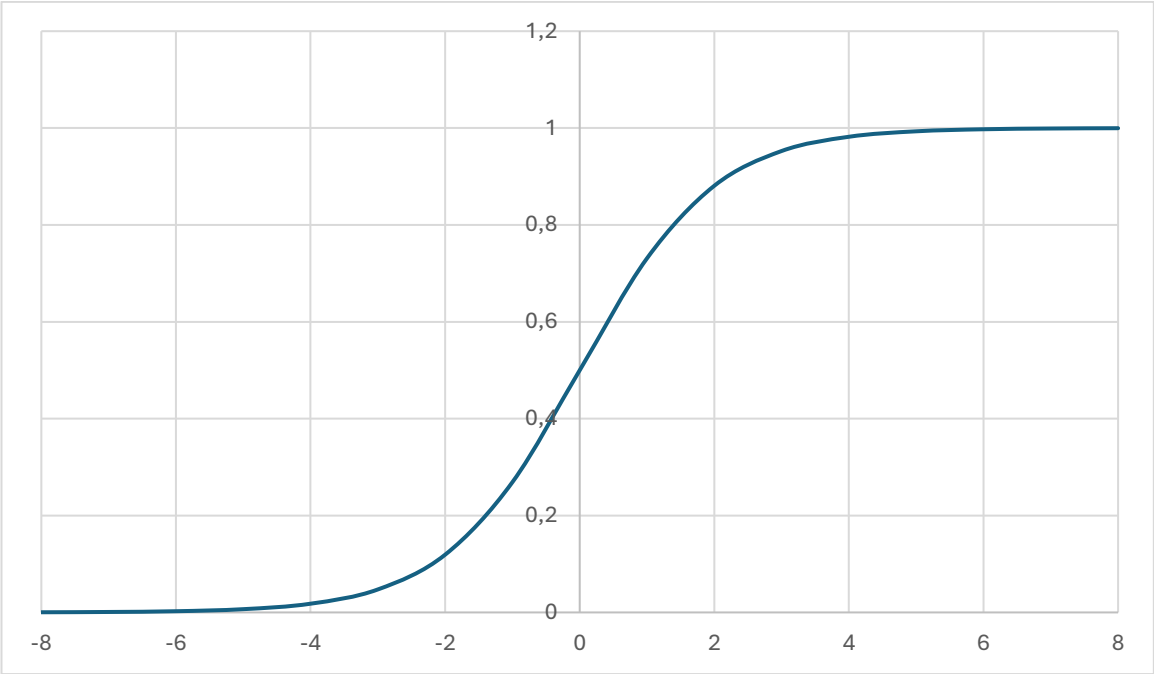

Рисунок 1. Сигмоида от X

Сигмоида переводит $X \in (-\infty; +\infty)$ в $Y \in (0; 1)$, то есть простыми словами, логистическая регрессия выдает вероятность принадлежности объекта к классу с меткой 1.

Теперь посмотрим, как из метода максимального правдоподобия получается оптимизационная задача, которую решает логистическая регрессия, а именно, – минимизация логистической функции потерь.

ММП:

$$\prod_{i=1}^{n} P(y_i|x_i, W) \to max$$

$$p_i = P(y_i = 1|x_i, W)$$

$$\prod p_i{}^{y_i}(1 - p_i)^{1-y_i} \to max$$

Логарифмируем:

$$\sum (\log (p_i)^{y_i} + \log(1 - p_i)^{1-y_i}) \to max$$

Применим свойство логарифма, домножим все выражение на -1 и усредним по количеству объектов:

$$L(W) = -\frac{1}{n}\sum (y_i \log(p_i) + (1 - y_i)\log(1 - p_i)) \to min$$

Данная функция отражает качество нашей модели: чем функция меньше, тем больше правдоподобие выборки, тем лучше модель.

Оптимизируется данная функция методом градиентного спуска. Посчитаем частную производную по весу w_j:

$$\frac{\partial L}{\partial w_j} = -\left(\frac{y_i}{p_i} - \frac{1 - y_i}{1 - p_i}\right)\frac{\partial p_i}{\partial w_j} =$$

$$-\left(\frac{y_i}{\sigma(< x_i, w >)} - \frac{1 - y_i}{1 - \sigma(< x_i, w >)}\right)\frac{\partial \sigma(< x_i, w >)}{\partial w_j} =$$

$$-\left(\frac{y_i}{\sigma(< x_i, w >)} - \frac{1 - y_i}{1 - \sigma(< x_i, w >)}\right)\sigma(< x_i, w >)\big(1 - \sigma(< x_i, w >)\big)x_{ij} =$$

$$-(y_i - \sigma(< x_i, w >))x_{ij}$$

Градиентный спуск заключается в том, чтобы итеративно вычитать градиент по весам, домноженный на *learning_rate*, из весов, тем самым приближаясь к локальному минимуму:

$$W_{t+1} = W_t - learning\_rate \times \nabla L(W_t)$$

Пример применения рассмотренного метода

Одно из главных преимуществ данного метода – интерпретируемость. Положительные веса говорят о том, что слово, соответствующее этому весу, является «спамным», отрицательные – об обратном.

Применим метод к датасету: https://github.com/pettum0104/tyunyatkin_kursovaya/blob/main/train_spam.csv

Описание датасета

Датасет состоит из двух колонок:

- text_type – является ли текст спамом или нет
- text – электронное письмо на английском языке

	text_type	text
0	ham	make sure alex knows his birthday is over in f...
1	ham	a resume for john lavorato thanks vince i will...
2	spam	plzz visit my website moviesgodml to get all m...
3	spam	urgent your mobile number has been awarded wit...
4	ham	overview of hr associates analyst project per ...
...	...	...
16273	spam	if you are interested in binary options tradin...
16274	spam	dirty pictureblyk on aircel thanks you for bei...
16275	ham	or you could do this g on mon 1635465 sep 1635...
16276	ham	insta reels par 80 गंद bhara pada hai 👀 kuch b...
16277	ham	alex s paper comments 1 in the sentence betwee...

16278 rows × 2 columns

Рисунок 2. Фрагмент датасета

Предобработка данных

Заменим метку ‘spam’ на единицу, ‘ham’ на ноль. Разделим выборку на обучающий и тестовый набор данных:

```
from sklearn.model_selection import train_test_split

train["text_type"] = train["text_type"].apply(lambda x: 1 if x=='spam' else 0)
X, y = train["text"], train["text_type"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((11394,), (4884,), (11394,), (4884,))
```

Рисунок 3. Предобработка данных

Проверим, нет ли явного дисбаланса классов во всей выборке и в тестовой:

```
y.value_counts()
```

```
text_type
0    11469
1     4809
Name: count, dtype: int64
```

```
y_test.value_counts()
```

```
text_type
0    3475
1    1409
Name: count, dtype: int64
```

Рисунок 4. Проверка наличия дисбаланса классов

Векторизация текстов

Векторизуем тексты с помощью CountVectorizer из библиотеки sklearn. Данный векторайзер реализует метод «Мешок слов».

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
X_train_vect = vectorizer.fit_transform(X_train)
X_test_vect = vectorizer.transform(X_test)
X_train_vect.shape, X_test_vect.shape
```

```
((11394, 43830), (4884, 43830))
```

Рисунок 5. Векторизация текстов

У нас получился словарь из 43830 уникальных слов.

Обучение логистической регрессии

Импортируем логистическую регрессию из библиотеки sklearn и обучим ее на обучающей выборке:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

clf = LogisticRegression(random_state=0)
clf.fit(X_train_vect, y_train)
predicted = clf.predict(X_test_vect)

print('Train accuracy:', accuracy_score(y_train, clf.predict(X_train_vect)),
      'Test accuracy:', accuracy_score(y_test, predicted))
```

```
Train accuracy: 0.9942074776197999 Test accuracy: 0.9443079443079443
```

Рисунок 6. Обучение логистической регрессии

Точность на тестовой выборке составила 94,4%.

Визуализация

```
y_val_proba = clf.predict_proba(X_test_vect)[:, 1]
token2id = {token: i for i, token in enumerate(vectorizer.get_feature_names_out())}
importance = clf.coef_[0]
min_importance = importance.min()
max_importance = importance.max()
TOKEN_PATTERN = "(?:\w|\')+"
ids = [6, 13, 23, 34, 35, 40]
for i in ids:
    review_body = X_test.iloc[i]
    print(f'spam: {bool(y_test.values[i])}')
    review_tokens = re.findall(TOKEN_PATTERN, review_body.lower())
    html_string = '''
    <p style="font-size:16px; color:#000000; border: 2px solid #000; text-align: justify; background-color:#ffffff;
    '''
    for token in review_tokens:
        if token in token2id:
            weight = importance[token2id[token]]
            if weight < 0:
                component = hex(int(255 - 255 * weight / min_importance))[2:]
                color = f'{component}{component}ff'
            else:
                component = hex(int(255 - 255 * weight / max_importance))[2:]
                color = f'ff{component}{component}'
        else:
            weight = 0.0
            color = 'ffffff'
        html_string += f'<span style="background-color: #{color}"; title="{weight:.2f}">{token}</span> '
    html_string += '</p>'
    display(HTML(html_string))
```

Рисунок 7. Код, отвечающий за визуализацию

Данный код выводит несколько текстов из тестовой выборки и раскрашивает спам-слова в красный цвет, слова, говорящие о том, что письмо не является спамом - в синий. Интенсивность цвета зависит от веса слова.

В результате получаем:

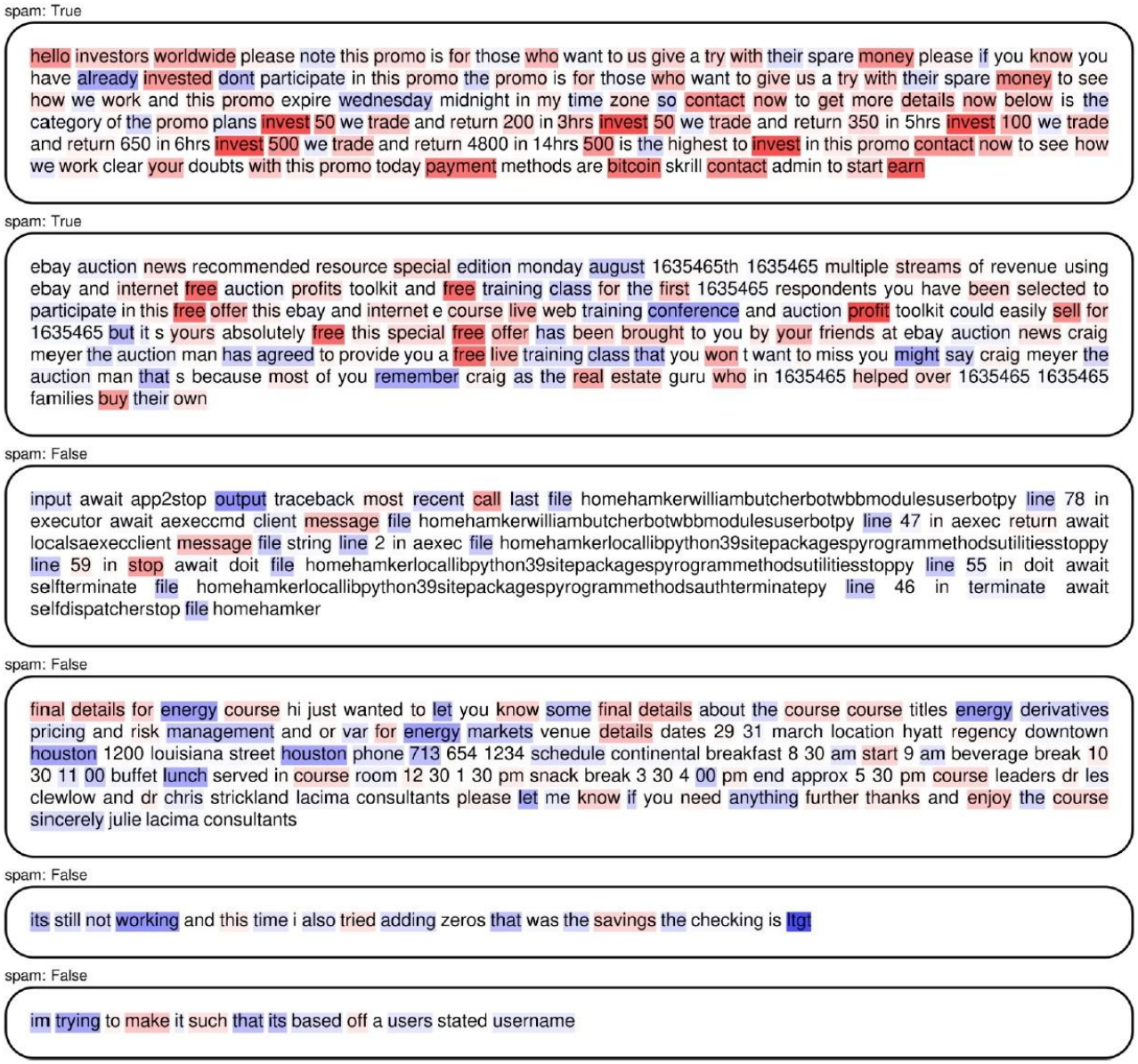

Рисунок 8. Визуализация

Видно, что самые «сильные» спам-слова: invest, earn, profit, free, что вполне логично. Антиспам-слова – обычные, привычные слова.

Заключение

В рамках данной курсовой работы был проведен комплексный анализ метода векторизации текстов "Мешок слов" и его применения в задачах антиспама. В ходе исследования была выведена и обучена логистическая регрессия, которая позволила оценить значимость различных слов для определения спам-сообщений.

Список используемых источников

1. Материалы курса Инженерно-математической школы НИУ ВШЭ и VK «Введение в анализ данных»
2. Материалы курса Инженерно-математической школы НИУ ВШЭ и VK «Машинное обучение»
3. Материалы курса Высшей школы экономики «Обработка и анализ данных»
4. Рашка, С. Python и машинное обучение / С. Рашка, В. Мирджалили

Результата антиплагиата

Заявка на проверку в системе Антиплагиат

Заявка № 1528193 от 27 Май 2024 г. 19:51:53

Заявитель: Тюняткин Пётр Александрович

На текущий календарный год у вас осталось 15 из 15 возможных проверок ваших работ в системе Антиплагиат.

Название работы:

Фишинг – актуальная угроза личности и организации в целом – цели, способы осуществления, классификация по урс

РЕЗУЛЬТАТЫ ПРОВЕРКИ

Статус:	Проверка закончена	Обновить статус
Доля заимствовани…	20,27	
Отчеты:	На сайте Антиплагиата	

Закрыть