МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования «Южно-Уральский государственный университет»

(национальный исследовательский университет)

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

Введение в обработку естественного языка

(название практической работы)

ОТЧЕТ

по практической работе № 1

по дисциплине «Интеллектуальные системы и технологии»

Выполнил:

студент группы КЭ–401

/ Д.А. Иванов /

(подпись)

« » 2024 г.

Проверил:

преподаватель кафедры СП

/ С.М. Абдуллаев /

(подпись)

« » 2024 г.

Оглавление

[1. Введение 3](#_Toc177662845)

[2. данные и методы 6](#_Toc177662846)

[3. результаты и анализ 9](#_Toc177662847)

[Литература 10](#_Toc177662848)

# Введение

В рамках работы по изучению подходов к обработке текстовых данных на примере задачи определения спама в смс сообщениях были освоены основные методы и технологии для обработки естественного языка.

Обработка текстов на естественном языке (Natural Language Processing, NLP) – общее направление искусственного интеллекта и математической лингвистики. Оно изучает проблемы компьютерного анализа и синтеза текстов на естественных языках. Применительно к искусственному интеллекту анализ означает понимание языка, а синтез – генерацию грамотного текста.

Для работы в данной области необходимо освоить основы языка программирования Python, а также таких библиотек как scikit-learn [1], pandas [2], numpy [3].

Обработка естественного языка (NLP) находит широкое применение в различных сферах. Она используется для создания чат-ботов и виртуальных помощников, понимающих и отвечающих на человеческую речь. NLP помогает в анализе настроений в социальных сетях, обнаружении спама и фейковых новостей, а также в машинном переводе. В бизнесе NLP применяется для автоматизации обслуживания клиентов, извлечения информации из текстов и персонализации маркетинговых кампаний. В научных исследованиях NLP используется для анализа больших объемов текстовых данных, например, научных статей или медицинских отчетов.

Одним из методов обработки текста является машинное обучение (ML). Это математико-статистические технологии, работающие с числами, поэтому перед непосредственной обработкой, необходимо пройти этапы предварительной обработки текста. К ним можно отнести такие подходы, как это удаление шума, стемминг и лемматизация. Под шумом подразумеваются стоп-слова – очень распространенные слова, которые не приносят особой пользы при анализе.

Лемматизация – это приведение измененных (или производных) слов к их основе, к базовой или корневой форме, которая представляет собой настоящие слова (леммы), в отличии от стемминга, который зачастую формирует не слова.

Для представления текстов в числовом виде применяются такие методы, как Мешок слов (Bag of Words) и TF-IDF. Мешок слов просто подсчитывает, сколько раз каждое слово встречается в тексте, игнорируя порядок слов. TF-IDF идет дальше и учитывает не только частоту слова в документе (TF), но и его важность для всего корпуса документов (IDF). Слова, которые встречаются часто в одном документе, но редко в других, считаются более информативными и получают больший вес. Таким образом, TF-IDF позволяет лучше выделить ключевые слова и улучшить качество анализа текста.

Чтобы преобразовывать слова в числовые векторы используют модель Word2Vec. Суть метода заключается в том, что слова, которые часто встречаются в схожем контексте, получают близкие по значению векторы. Это позволяет модели уловить семантические и синтаксические отношения между словами. Процесс преобразования текста в числовой вектор с помощью Word2Vec выглядит следующим образом: сначала строится словарь всех слов в корпусе данных, затем каждое слово представляется в виде уникального вектора высокой размерности (например, 300). В ходе обучения модели, эти векторы настраиваются таким образом, чтобы максимизировать вероятность предсказания контекстных слов на основе целевого слова или наоборот. В результате, мы получаем векторное представление текста, которое можно использовать для различных задач NLP, таких как классификация текстов, кластеризация, поиск синонимов и антонимов.

Для анализа получившихся векторов используют различные метрики, например, косинусную близость. Она вычисляется как косинус угла между двумя векторами. Если угол между векторами мал, то их косинусная близость близка к 1, что означает высокую степень сходства. Напротив, если угол большой, то косинусная близость близка к 0, что указывает на низкую степень сходства.

Еще одним средством анализа текста является установление взаимосвязи между словами в предложении, или парсинг зависимостей. Самыми популярными инструментами для парсинга зависимостей на сегодняшний день являются SpaCy, Stanford CoreNLP, NLTK, UDPipe, AllenNLP. Выбор программы зависит от конкретных задач, языка, доступных ресурсов и требуемой точности.

# данные и методы

Для обработки будем использовать корпус опубликованный в открытом источнике [5]. Он представлен в листинге 1.

Листинг 1 – Корпус для анализа

corpus = [

"Tune a hyperparameter.",

"You can tune a piano but you can't tune a fish.",

"Fish who eat fish, catch fish.",

"People can tune a fish or a hyperparameter.",

"It is hard to catch fish and tune it.",

]

Метод Bug of Words реализован с помощью класса CountVectorizer в библиотеке scikit-learn. Пример использования данного класса представлен в листинге 2.

Листинг 2 – Генерация Bug of Words с помощью CountVectorizer

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

vectorizer = CountVectorizer(stop\_words='english')

features = vectorizer.get\_feature\_names\_out()

X = vectorizer.fit\_transform(corpus)

df = pd.DataFrame(X.toarray(), columns=features)

print(df)

Результат работы данного кода представлен в таблице 1.

Таблица 1 – Матрица Bag of Words

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | catch | eat | fish | hard | hyperparameter | people | piano | tune |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 |
| 2 | 1 | 1 | 3 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 4 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

Метод TF-IDF реализован с помощью класса TfidfVectorizer в библиотеке scikit-learn. В листинге 3 представлен пример кода, для генерации матрицы с ТF-IDF. Заметим, что в аргументах класса указана опция *use\_idf=False*, которая настраивает векторайзер на расчет TF (Term Frequency) без IDF, что является нормализированной формой Bag of Words.

Листинг 3 – Генерация TF с помощью TfidfVectorizer

import pandas as pd

import numpy as np

from sklearn.feature\_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop\_words='english', use\_idf=False)

X = vectorizer.fit\_transform(corpus)

features = vectorizer.get\_feature\_names\_out()

data = np.round(X.toarray(), 3)

df = pd.DataFrame(data, columns=features)

print(df)

Результат представлен в таблице 2.

Таблица 2 – Матрица TF

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | catch | eat | fish | hard | hyperparameter | people | piano | tune |
| 0 | 0.000 | 0.000 | 0.000 | 0.000 | 0.707 | 0.000 | 0.000 | 0.707 |
| 1 | 0.000 | 0.000 | 0.408 | 0.000 | 0.000 | 0.000 | 0.408 | 0.816 |
| 2 | 0.302 | 0.302 | 0.905 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 3 | 0.000 | 0.000 | 0.500 | 0.000 | 0.500 | 0.500 | 0.000 | 0.500 |
| 4 | 0.500 | 0.000 | 0.500 | 0.500 | 0.000 | 0.000 | 0.000 | 0.500 |

Применим к получившемуся результату IDF (Inverse Document Frequency). Для этого необходимо убрать опцию *use\_idf=False* из аргументов класса. Код представлен в листинге 4.

Листинг 4 – Генерация TF-IDF с помощью TfidfVectorizer

import pandas as pd

import numpy as np

from sklearn.feature\_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop\_words='english')

X = vectorizer.fit\_transform(corpus)

features = vectorizer.get\_feature\_names\_out()

data = np.round(X.toarray(), 3)

df = pd.DataFrame(data, columns=features)

print(df)

Результат представлен в таблице 3.

Таблица 3 – Матрица TF-IDF

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | catch | eat | fish | hard | hyperparameter | people | piano | tune |
| 0 | 0.000 | 0.000 | 0.000 | 0.000 | 0.820 | 0.000 | 0.000 | 0.573 |
| 1 | 0.000 | 0.000 | 0.350 | 0.000 | 0.000 | 0.000 | 0.622 | 0.701 |
| 2 | 0.380 | 0.471 | 0.796 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 3 | 0.000 | 0.000 | 0.373 | 0.000 | 0.534 | 0.661 | 0.000 | 0.373 |
| 4 | 0.534 | 0.000 | 0.373 | 0.661 | 0.000 | 0.000 | 0.000 | 0.373 |

# результаты и анализ

Во всех сгенерированных матрицах строками являются предложения из корпуса (документы), столбцами – слова.

В Bag of Words (таблица 1) в ячейках содержится частота использования отдельного слова. Видно, что в документе под номером 2, слово «fish» встречается чаще всего в предложении: 3 раза.

Принцип работы TF схож с Bag of Words, однако частота слов нормализована: разделена на длину вектора. В документе под номером 0 видно, специфичное слово «hyperparameter» имеет одинаковый вес с распространенным словом «tune. В третьем документе слово «fish» имеет большое значение 0.905, поскольку встречается целых 3 раза.

В случае с TF-IDF (таблица 3) ситуация резко меняется. IDF добавляет вес специфичным словам, которые встречаются реже, чем остальные. Заметно, что значение слова «hyperparameter» в документе 0 выросло, так как оно редко фигурирует во всем корпусе. Также можно заметить, что специфичные для предметной области слова имеют больший вес, чем слова с высокой частотой.

# Литература

1. Официальная страница библиотеки Scikit-Learn [Электронный ресурс] URL: https://scikit-learn.org/
2. Официальная страница библиотеки Pandas [Электронный ресурс] URL: https://pandas.pydata.org/
3. Официальная страница библиотеки NumPy [Электронный ресурс] URL: https://numpy.org/
4. Официальная страница библиотеки NumPy [Электронный ресурс] URL: https://numpy.org/
5. Статья «A Quick Introduction to Bag of Words and TF-IDF» [Электронный ресурс] URL: https://dataknowsall.com/blog/bowtfidf.html