ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА – СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“

Курс Системи, основани на знания
за специалност Информационни системи
зимен семестър 2021/2022 г.

Разпознаване (класификатор) на авторство

курсов проект

Мария Андреева Качаунова
Факултетен номер 71996
Петър Димитров Пивчев
Факултетен номер 72009

Дата
януари 2022г.

Съдържание

1. Увод

a) Описание и идея на проекта

Идеята на проекта е да се реализира програма, която представлява разпознаване, т.нар. класификатор, на авторство, чиято цел е разпознаването текстове на Иван Вазов от текстове на Йордан Йовков. Разпознаването на авторство е предмет на много проучвания, тъй като с развитието на Интернет, нараства и обема на разпространение на анонимна информация, плагиатство, а също е и използвано средство за разпознаване на автори на исторически документи.

b) Цел и задачи на разработката

Избиране на подходящ алгоритъм за решаване на задачата, извличане на входни данни от текстовете.

c) Структура на документацията

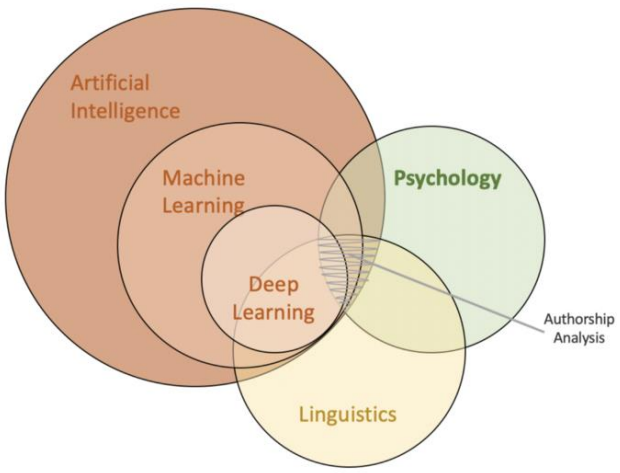
В документацията на този проект се описват идеята, целите, крайният резултат и трудностите, срещнати по време на изпълнението. Описва се структурата на проекта, взетите решения и са включени извадки от кода на програмата.

2. Преглед на предметната област

a) Основни дефиниции и концепции

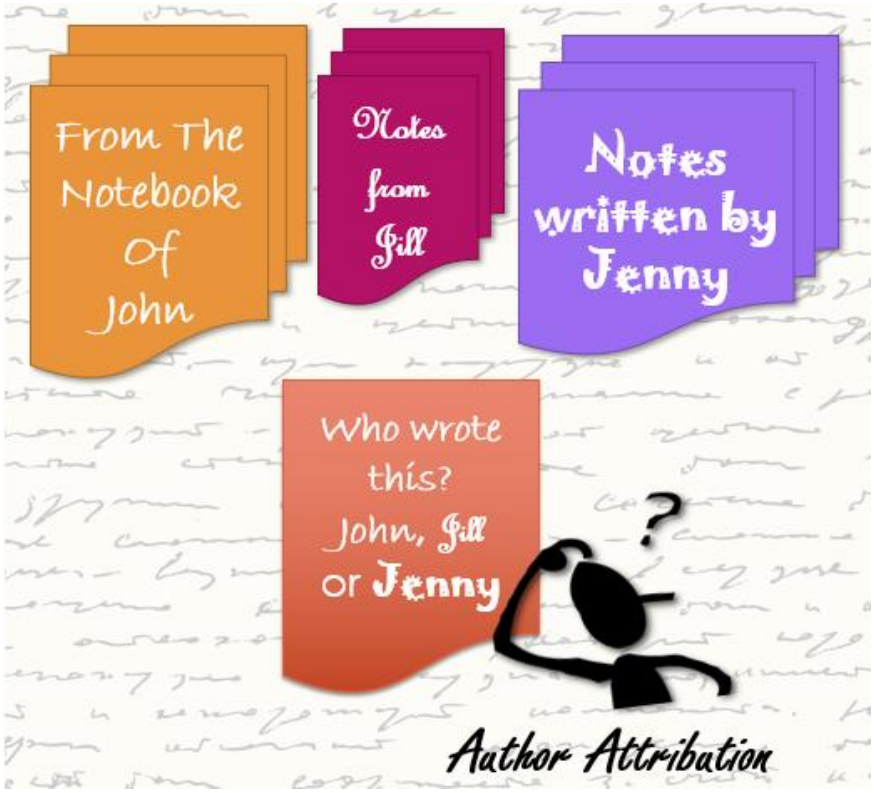
i. *Authorship Analysis*

Изкуството и науката за разграничаване на стиловете на писане на авторите чрез идентифициране на характеристиките на личността на авторите и разглеждане на статии, създадени от тях, се нарича *Анализ на авторството*.



ii. ***Author Attribution***

Разпознаването (приписване) на автора определя, че след проучване на текст от колекция от множество автори с недвусмислено авторство, ако непредвиден текст е написан от конкретно лице. В идеалния случай това е проблем за класификация на текст от затворен набор от няколко класа. Това е процеса за определяне на автор на даден текст, чрез представяне на текстове, чийто автор е ясен.



Разпознаването на авторството е един проблем с класификацията, при който освен текстото съдържание, е важен и стилът на писане на автора.

iii. ***Стилометрия***

Стилометрия е приложението на изучаването на лингвистичен стил, обикновено към писмения език. Статистическият анализ на стила, стилометрията, се основава на предположението, че всеки авторски стил има определени черти, достъпни за съзнателно манипулиране. Характеристиките, които са специфични за стилометрията, са както следва: брой изречения в

статия, брой думи в статия, среден брой думи в изречение, средна дължина на думата в статия, размер на речника на автора (богатство на думи), брой на точки, брой удивителни знаци, брой запетаи, брой двоеточия, брой точки и запетая, брой непълни изречения. Тези характеристики могат да се използват заедно с един класификатор.

b) **Описание на използвания алгоритъм за решаване на задачата**

i. За решаването на дадения проблем използваме тип машинно самообучение, ***машинното самообучение с учител*** или *supervised machine learning*, или по-точно самообучение чрез запомняне. Това е подход за самообучение с учител, при който целта е на базата на множество, зададени от учителя решени примери, т.нар. *обучаващи примери*, за обекти, принадлежащи на множество от известни класове, да се класифицира определен тестов пример.

ii. ***K-nearest neighbour algorithm***

Най-използваният метод за самообучение чрез запомняне е методът на най-близкия съсед (*NN – Nearest Neighbour*). При този метод класификацията на тестовия пример зависи от степента на неговото сходство с единствен пример на понятие (клас) – този, който се намира на най-малко разстояние от него (неговия най-близък съсед).

Това е непараметричен и мързелив алгоритъм, т.е. той прави избора си въз основа на близостта до други точки от данни, независимо от характеристиката на числовите стойности и има малка или никаква фаза на обучение.

Положителните страни на този алгоритъм са, че не прави предположения относно данните, лесен за разбиране е и може да се ползва за класификация и регресия. От друга страна, изисква много памет и е чувствителен към неподходящи характеристики и към мащаба на данните, тъй като се изчислява разстоянието до най-близките *k*-точки.

iii. ***Gaussian Naive Bayes Classifier***

Едно от най-полезните приложения на правилото на Бейс е т.нар. наивен Бейсов класификатор.

Бейсовият класификатор е техника за машинно самообучение, която може да се използва за класифициране на обекти, например текстови документи, в два или повече класове. Класификаторът се обучава чрез анализиране на

набор от данни за обучение, за които са зададени правилните класове.
Gaussian naive Bayes алгоритъма е специален тип наивен Бейсов класификатор. Използва се специално, когато характеристиките имат непрекъснати стойности. Също, се предполага, че всички характеристики следват гаусовото разпределение, т.е. нормалното разпределение.

iv. ***Логистична регресия***

Логистичната регресия е алгоритъм за машинно обучение, който се използва за проблемите с класификацията, това е алгоритъм за прогнозен анализ и се основава на концепцията за вероятност. Използва се за приписване на наблюдения към дискретен набор от класове. Някои от примерите за проблеми с класификацията са имейл спам или не, онлайн транзакции измама или не**.** При логистичната регресия използваните данни могат да бъдат качествени или количествени, но резултатът винаги е категоричен.

3. Проектиране

а) Описание на структурата и произхода на използваните данни

<u>Първа стъпка.</u>

Събиране и извличане на данни. „*Изчистване*“ на данни. В тази стъпка се извличат данните, които ще ползваме за обучаващи примери.
Идеята е да вкараме всички текстове на даден автор в един стринг след което ги „*изчистваме*“ по символи. Прилагаме операции по „*изчистването*“, т.е. премахваме пунктуационни знаци, големите букви стават малки и се премахват думи и поредици от символи, които са различни от кирилицата. След това тези данни се вкарват в масив, който съдържа две колони – едната, съдържа името на автора, а другата съответните му текстове.

Използваните текстове са:
- Иван Вазов – „Под игото“ и „Епопея на забравените“ – 129, 847 символа
- Йордан Йовков – „Чифликът край границата“, „Песента на колелетата“, „Приключенията на Гороломов“, Старопланински легенди“ – 196,328 символа

Първа стъпка е разработена, използвайки *java*. От нея получаваме два *.txt* файла – *vazovWords.txt* и *jovkovWords.txt.*

```
public static String readFromFile(String filePath)
{
    try
    {
        File file = new File(filePath);
        Scanner scanner = new Scanner(file);
        int counter = 0;
        ArrayList<String> data = new ArrayList<>();
        while (scanner.hasNextLine())
        {
            data.add(counter, scanner.nextLine());
            ++counter;
        }
        scanner.close();

        return data.toString();
    }
    catch (FileNotFoundException e)
    {
        System.out.println("ERROR");
        e.printStackTrace();
        return null;
    }
}
```

Четем различните текстове и ги прибавяме към *ArrayList<String>*.

```
public static String transformAndTokenize(String rawText)
{
    return rawText.replaceAll( regex: "[^а-яА-Я\\s]", replacement: "").toLowerCase(Locale.ROOT).
            replaceAll( regex: "\\s+", replacement: " ");
}
```

Премахваме пунктуационни знаци, големите букви стават малки и се премахват думи и поредици от символи, които са различни от кирилицата.

```
public static void writeToFile(String path, String text)
{
    try
    {
        File file = new File(path);
        FileWriter fw = new FileWriter(file);
        PrintWriter pw = new PrintWriter(fw);

        pw.print(text);

        pw.close();

    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Записваме „изчистените“ данни в нов *.txt* файл. С други думи, сериализираме във файл чрез *String.*

```
String text = Authorship.transformAndTokenize(
        Objects.requireNonNull(Authorship.readFromFile(
                filePath: "C:\\Users\\Legion\\IdeaProjects\\regextest\\" +
                        "src\\AuthorshipAttribution\\jovkov\\" +
                        "Jordan_Jovkov_-_Staroplaninski_legendi_-_522-b.txt"
        ))
);

Authorship.writeToFile( path: "C:\\Users\\Legion\\IdeaProjects\\regextest\\src\\jovkovWords.txt", text);
```

```
public static void appendToFile(String path, String text)
{
    try
    {
        File file = new File(path);
        FileWriter fw = new FileWriter(file,  append: true);
        PrintWriter pw = new PrintWriter(fw);

        pw.print(text);

        pw.close();

    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Ползваме тази функция ако вече има наличен файл с текстове, за да се избегне презаписването.

Използваме *python*, за да използваме библиотеките за *lemmatization* и *stemming*, за да може да трансформираме думите, т.нар. *tokens*, към техните корени.

```
stop_words = get_stop_words('bulgarian')
lemmatizer = Lemmatizer('bg')
stemmer = BulStemmer.from_file('AuthorshipAttribution/raw words tokenized/bulstem_bg.txt',
                               min_freq=2, left_context=2)

#splits the text on whitespaces
tokens = text.split()

#removes stopWords aka words like is, with, up, etc and other non-word tokens
tokens = [token for token in tokens if token not in stop_words and
          all(c.isalpha() for c in token)]

#Using lemmatization to go from 'македонският' to 'македонски'
tokens = [lemmatizer.lemmatize(token) for token in tokens]
#Using stemming to go from 'македонски' to 'македонск'
tokens = [stemmer.stem(token) for token in tokens]
```

Втора стъпка.

Извличане на характеристики. След събирането и „*изчистването*“ на данните, трябва да бъдат намерени отличителни характеристики, чрез които могат да бъдат разграничени авторите.

При *класификаторът на автори* е от значение не само съдържанието на текста, а и стилометрията и други качества, които разграничават характеристиките на даден автор.

Използваме стилометрията, речниковото разнообразие, честота на използване на функционални думи (частици, местоимения, съюзи).

Методи за представяне на информацията – идеята тук е на всяка дума да бъде дадена оценка и съставяме в табличен вид статистика за думата с най-висока тежест и нейния автор, и за думата с най-висока средна тежест и нейния автор.

Трета стъпка.

Създаваме модел, чрез използване на логистична регресия. Първо избираме най-подходящ модел сред *logistic regression, K-NN method, Gaussian Naive Bayes Classifier*, чрез използване на k-итеративни разделяния. След използване на модел, който реализира логистична регресия намираме думите, които най-много отличават двамата автори.

Четвърта стъпка.

Тук целта ни е да се провери и оцени работата на модела като използваме случайно подбрани части от текстове, за да проверим кой е техният автор.

Формира се случайна извадка от всеки текст и за всяка случайна такава се избира случайна поредица от символи и прилагаме модела върху нея.

Методът се оценява като:

- намираме процент коректни класификации
- генерираме матрица за разбъркване на обърквaнето и се генерират произхождащи от нея метрики

4. Реализация на проекта

а) Примерни резултати от работата на използваното приложение

```
Authorship counterOfWords = new Authorship(
        Authorship.transformAndTokenize(
                Objects.requireNonNull(Authorship.readFromFile(
                        filePath: "C:\\Users\\Legion\\IdeaProjects\\regextest\\src\\jovkovWords.txt"
                ))
        )
);

System.out.printf("%s: %d\n", "Number of words in Jordan Jovkov's texts", counterOfWords.getData().length);

System.out.printf("%s: %d\n","Number of characters in " +
        "Jordan Jovkov's texts", (Objects.requireNonNull(Authorship.readFromFile(
                filePath: "C:\\Users\\Legion\\IdeaProjects\\regextest\\src\\jovkovWords.txt"
        )).length()));
```

```
Number of words in Jordan Jovkov's texts: 196328
Number of characters in Jordan Jovkov's texts: 1073872

Number of words in Ivan Vazov's texts: 196328
Number of characters in Ivan Vazov's texts: 743053
```

5. Заключение

По темата на проекта има множество проучвания, които ползват различни алгоритми за решаването на проблема като процента на успеваемост при всеки един е различен, в зависимост от стила, сферата, времето на писане и тн. на текстовете.

По време на разработката на този проект срещнахме трудности, основно свързани, с непознаването на подходящи езици за решаване на проблеми от такъв тип. В бъдеще, добра реализация би била да се ползва само един език и също, да бъде улеснено събирането на подходящите текстове за обработка на данни, както и подходящ интерфейс за потребителя.

6. Източници

i. towardsdatascience.com
ii. dataaspirant.com
iii. https://users.cs.duke.edu/~ilker/papers/conference/iscs07.pdf
iv. https://bg.differencevs.com/
v. bg.wikipedia.org
vi. https://bg.memberpedia.net/
vii. elementsofai.com
viii. https://www.youtube.com/watch?v=XFSzCVrn-dU
ix. https://www.youtube.com/watch?v=DU8I8B8uNgA