Класификация на медицински текст

Документация на проект за курса по
“Подходи за обработка на естествен език”
на Факултета по Математика и Информатика
към Софийски университет “Св. Климент Охридски”

Автор: Ася Русанова
ФН: 3MI3400047
Магистърска програма, Курс 1
Специалност: Изкуствен Интелект

1. Увод

1.1. Мотивация

Около 30% от генерираните данни по света са свързани с областта на здравеопазването. Неструктурираността на този вид данни (било то лекарски бележки или описание на симптоми директно от пациента) води до отежняване на процеса по извличане на валидна медицинска информация. Този проект има за цел да рационализира процеса по извличане на данни, използвайки NLP-базиран инструмент за класификация на текст. Основната задача на проекта се изразява в правилното класифициране на описани от пациенти симптоми в няколко категории.

1.2. Език за програмиране и използвани библиотеки

Проектът е реализиран на програмния език Python поради наличието на редица полезни функционалности за обработка на лингвистични данни. Ключова роля в разработката играе библиотеката на Python NLTK, посредством която са реализирани някои от основните догми в обработката на текст - лематизация, премахване на стоп думи, токенизация и т.н. За целите на създаването на модела е включена библиотеката ski-kit learn (sklearn), съдържаща редица ефективни инструменти за машинно самообучение и статистическо моделиране. За извличане на оригиналното множество от данни е използвана библиотеката pandas, а библиотеките seaborn и matplotlib са използвани за визуализационни цели.

Проектът е реализиран с помощта на *Google Colab* и може да бъде намерен на следния адрес : https://colab.research.google.com/drive/1AhCKc6cy1Pca8CUvz0MYQbKokq5eMf8c?usp=sharing

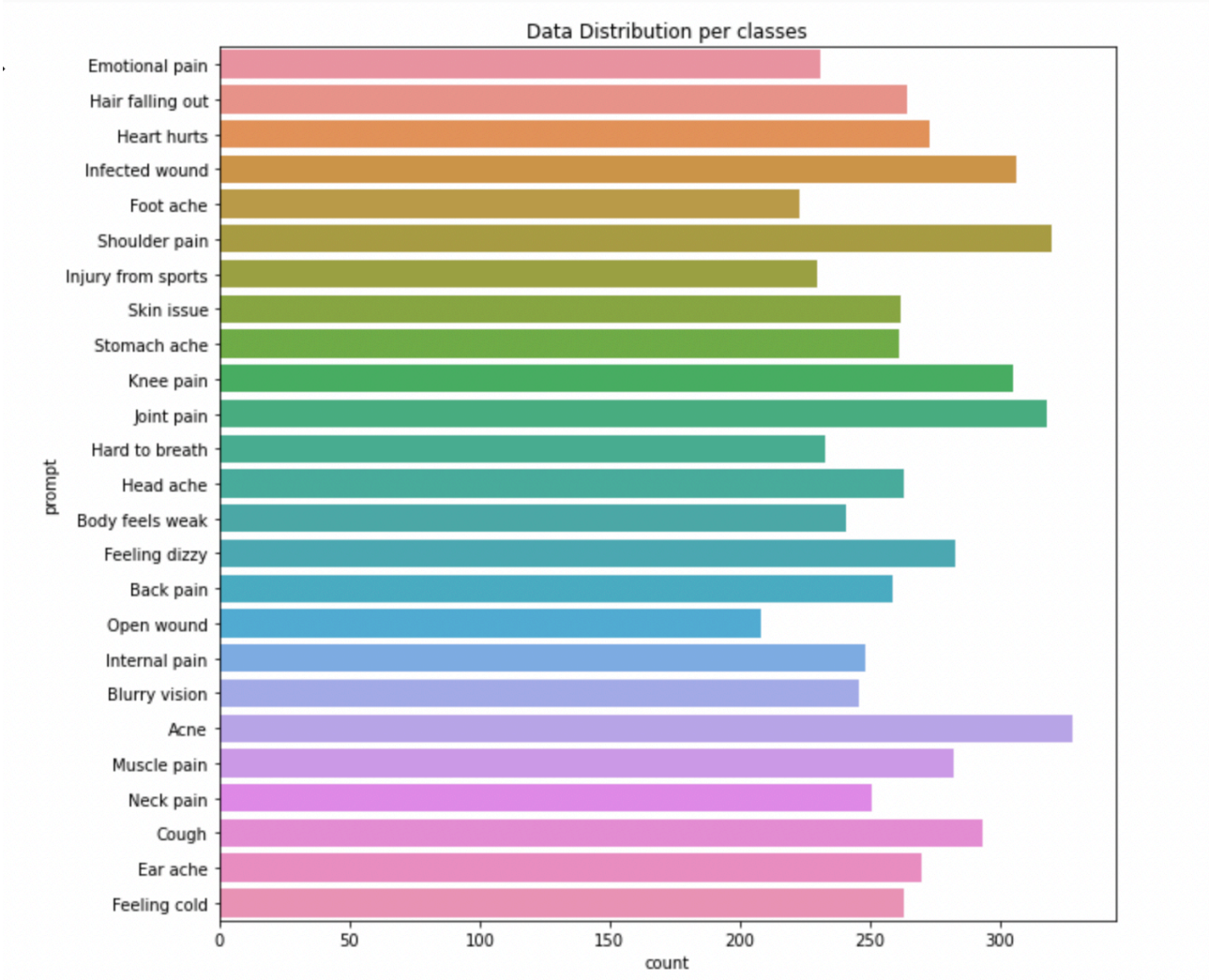
2. Данни - Анализ

Множеството от данни, на което се базира проекта, може да бъде намерено в платформата Kaggle посредством следния линк: https://www.kaggle.com/paultimothymooney/medical-speech-transcription-and-intent. Смислово, данните могат да бъдат разделени на две подкатегории спрямо формата си. Първата подкатегория от данни се състои от аудио записи на пациенти, споделящи своите симптоми. Аудио записите са в .wav формат и общата им продължителност надхвърля 8 часа. Направена е детайлна транскрипция на тези аудио данни, която е съхранена в .csv файл, именован '*overview_of_recordings.csv*'. Втората подкатегория от данни е именно този файл. Файлът съдържа голямо количество метаданни като качество на записа, наличие на фонов шум, озвучаване на пациента, име на аудио файла, номер на пациента, номер на писателя (човека, направил транскрипцията) и т.н.. Две са колоните, които представляват интерес за класификационната задача - *phrase*, съдържаща транскрипцията на аудио файла на даден пациент, изказващ оплакванията си, и prompt - класификацията на оплакванията с конкретен симптом.

Данните в '*overview_of_recordings.csv*' файла са извлечени посредством библиотеката pandas. Общият брой данни след премахване на редовете, съдържащи NaN стойности, е 6,661. Проучването върху данните се съсредоточава върху колоните *phrase* и *prompt* от .csv файла и игнорира останалите метаданни, които не оказват влияние на класификацията. Броят различни стойности в колоната *prompt*, съответстващ на броя различни класове в задачата, е 25. Пример за клас е 'Knee pain', а пример за елементи,

принадлежащи към този клас, са изреченията: ‘I have knee pain when I walk a lot’ (ред: 13), ‘My knee catches and hurts when I first stand up after sitting’ (ред: 63) и т.н. Общият брой изречения, асоциирани с класа ‘Knee pain’, е 305.

Направен е анализ на разпределението на елементи в отделните класове, като е стигнато до заключението, че данните са балансирани и следователно няма нужда от допълнителна регуляризация. На всеки клас съответстват между 250 и 300 фрази от множеството от данни. Визуално, балансираността на данните може да се види в Таблица 1.



Фигура 1.

3. Предварителна обработка на множеството от данни

Предварителната обработка на данните е систематизирана в три етапа - обработка на фразите, обработка на класовете и разделяне на данните на множество за обучение и тестово множество.

3.1. Обработка на фразите

За обработката на фразите са използвани няколко техники за почистване на текст, които са разделени в следните 3 функции:

- `clean_phrase(text)` - отговаря за премахване на пунктуационни символи и цифри, както и за конвертиране на изреченията в изречения, състоящи се само от малки букви
- `remove_stopwords_from_phrase(text)`- отговаря за премахване на така наречените "стоп-думи" от текста. Стоп-думите обикновено се отнасят до най-често срещаните думи в даден език (например "and", "the", "a"). При прилагане на модели за машинно самообучение към текст, тези думи могат да добавят много шум, поради което възниква и нуждата от тяхното отстраняване. NLTK библиотеката има предварително дефиниран списък от стоп-думи, съдържащ най-често срещаните думи в един език. За целите на задачата са използвани вградените в NLTK стоп-думи за английски език.
- `lemmatize_phrase(text)` - извършва лематизация. Лематизацията е процес, който съпоставя на различните форми на една дума каноничната форма на думата (известна още като лексема или лема). За реализацията на този метод отново е използвана NLTK библиотеката.

Трите функции за обработка на фразите са приложени последователно върху всяка фраза от множеството.

3.2. Обработка на класовете

В оригиналното множество от данни класовете са представени посредством символни низове. С цел по-лесна обработка е извършено съпоставянето клас - етикет (число от 0 до 24). Резултатното множество от етикети кореспондира на следното множество от класове, подредено в азбучен ред (вж. Фигура 2.)

```
['Acne' 'Back pain' 'Blurry vision' 'Body feels weak' 'Cough' 'Ear ache'
'Emotional pain' 'Feeling cold' 'Feeling dizzy' 'Foot ache'
'Hair falling out' 'Hard to breath' 'Head ache' 'Heart hurts'
'Infected wound' 'Injury from sports' 'Internal pain' 'Joint pain'
'Knee pain' 'Muscle pain' 'Neck pain' 'Open wound' 'Shoulder pain'
'Skin issue' 'Stomach ache']
```

Фигура 2.

Пример за фраза преди и след нейната обработка, както и за класа и етикета, с който е класифицирана фразата е показан на Фигура 3.

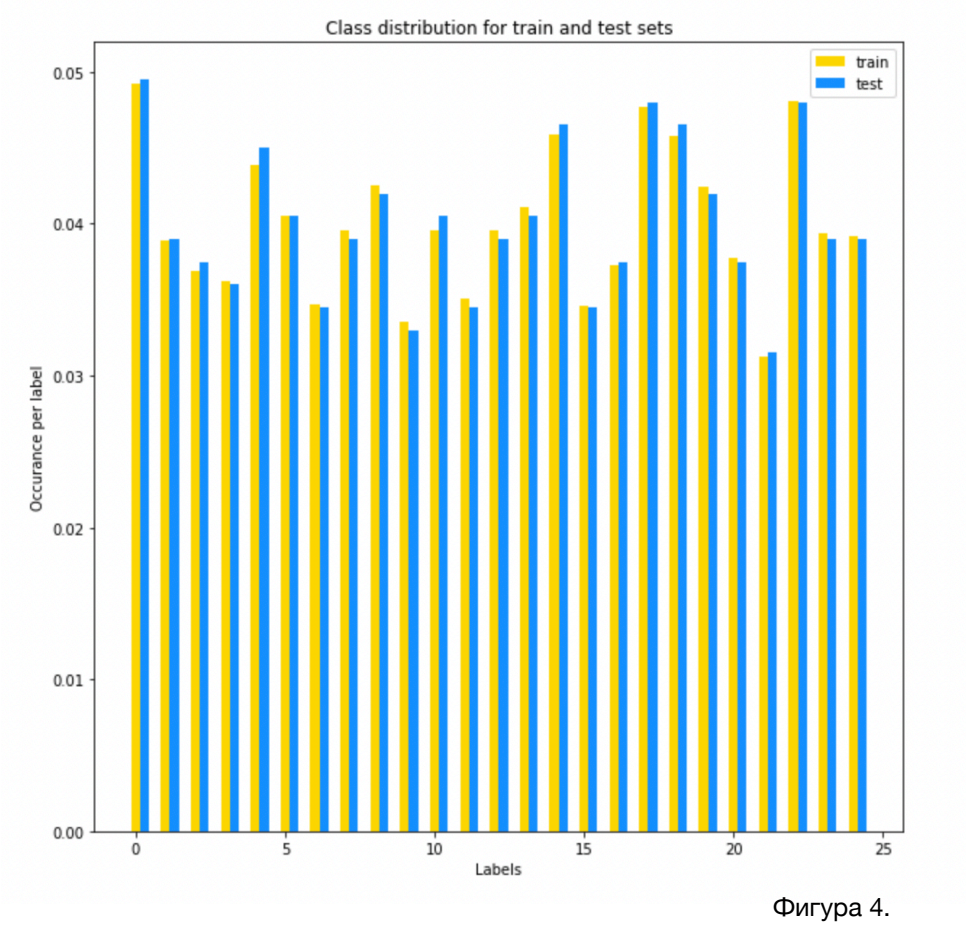
```
Phrase: I feel dizzy when I set in-front of my laptop for an hour or two, what possibly could be the reason?
Pre-Processed Phrase: feel dizzy set infront laptop hour two possibly could reason
Prompt: Feeling dizzy
Corresponding prompt label: 8
```

Фигура 3.

3.3. Разделяне на данните на обучаващо и тестово множество

След предварителната обработка на фразите и етикетирането на класовете е извършено разделяне на данните на множество за обучение и множество за тестване със съотношение 9:1. За тази цел е използван метода `train_test_split()`на библиотеката sklearn, като с цел балансиране на броя екземпляри във всеки клас от двете новосъздадени множества е използван атрибута `stratify`.

Доказателство за балансираният брой екземпляри от всеки клас в обучаващото и тестовото множество предоставя Фигура 4.



Фигура 4.

4. TF-IDF

Преди прилагането на модела за машинно самообучение, фразите (на обучаващото и тестовото множество) се конверитрат във вектори от числа. На всяка дума от една фраза е съпоставено число, съответстващо на важността на думата в конкретната фраза, като за оценка на важността е използвана статистическата метрика TF-IDF. Стойността на оценката на TF-IDF за една дума нараства пропорционално с броя срещания на думата в конкретната фраза, но се балансира с оглед на броя фрази от масива от фрази, които съдържат думата.

Стойността на оценката на TF-IDF за думата x във фразата y: $w_{x,y}$ се изчислява по формулата:

$w_{x,y} = tf_{x,y} \times log(\frac{N}{df_x})$, където

$tf_{x,y}$ е броят срещания на думата x във фразата y,

df_x е броят срещания на думата x в целия корпус от фрази,

N е общият брой фрази в корпуса.

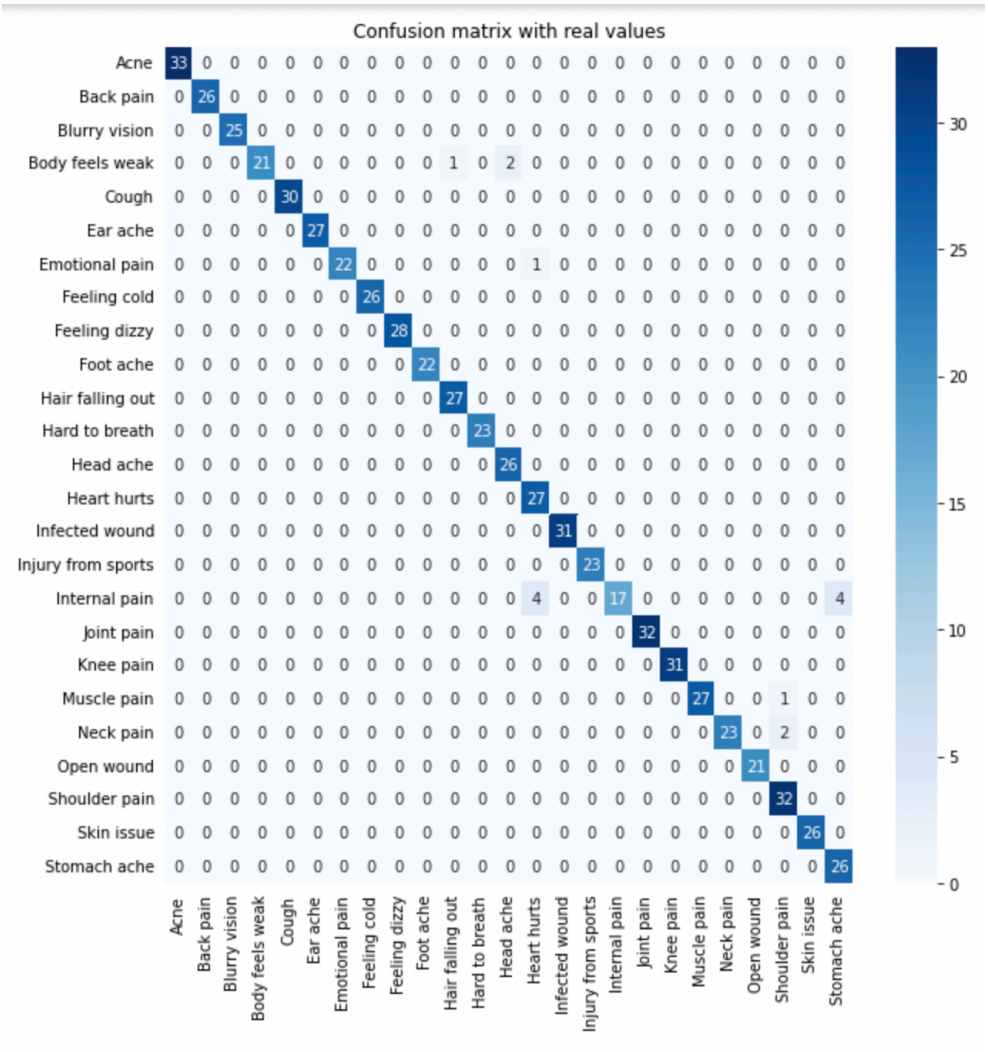
С цел извършване на векторизацията на фразите посредством статистическата метрика TF-IDF се използват две функции на `TfidfVectorizer`. За векторизация на обучаващите фрази се използва метода `fit_transform()`, който мащабира тренировъчните данни и също научава параметрите за мащабиране на тези данни, докато за векторизация на тестовите фрази се използва метода `transform()`. Посредством този метод, за трансформиране на тестовите данни се използват изчисленията от данните за обучение.

5. Класификационен модел

За класификатор в задачата е използван наивен Бейсов класификатор. Наивният Бейсов класификатор се опитва да изчисли вероятността за срещане на определен клас c при наличието на поредица от думи w_i в дадена фраза: $P(c \mid w_0, \ldots, w_n)$.
За опростяване на тази условна вероятност се използва правилото на Бейс, както и предположението, че срещаните думи са независими една от друга.
За апроксимация на вероятността $P(w_i \mid c)$ се използва MultinomialNB.

6. Резултати

Постигнати са добри резултати с точност над тестовото множество от порядъка на 97%. За изследване на резултатите е използвана т.нар. confusion matrix, показана на Фигура 5. По диагонала на матрицата се намират правилно класифицираните фрази. Най-много грешки се наблюдават във фрази от класа 'Internal pain', като 4 фрази от този клас са погрешно класифицирани като принадлежащи на класа 'Heart hurts', а други 4 фрази от този клас са погрешно класифицирани като принадлежащи на класа 'Stomach ache'.



Фигура 5.

7. Използвана Литература

(1) Лекции по “Подходи за обработка на естествен език”, Г. Ангелова

(2) “Natural Language Processing with Python”. S. Bird, E. Klein, E. Loper

(3) “Python for Data Analysis”, W. McKinney

(4) NLTK Documentation: https://www.nltk.org

(5) SkLearn Documentation: https://scikit-learn.org/0.21/documentation.html

(6) Pandas Documentation: https://pandas.pydata.org/docs/index.html

(7) TF-IDF: https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3

(8) TF-IDF: https://monkeylearn.com/blog/what-is-tf-idf/

(9) Difference between fit_transform() and transform() : https://towardsdatascience.com/what-and-why-behind-fit-transform-vs-transform-in-scikit-learn-78f915cf96fe