System oceny przydatności pytań na podstawie ich meta danych i treści dla portalu StackOverflow w celu ułatwienia pracy moderatorom

Dokumentacja projektu

Zespół

- Dański Mateusz
- Pietrzak Mateusz

Spis treści

Motywacja

StackOverflow to platforma, która pozwala użytkownikom rozwiązywać problemy z kodem aplikacji, wyjaśnić zagadnienie informatyczne itp. poprzez zadawanie pytań. Użytkownicy posiadający odpowiednią reputację odpowiadają na pytania, a po znalezieniu najlepszej odpowiedzi pytanie zostaje zamknięte przez pytającego. Niestety większość pytań wciąż pozostaje otwarta, albo zostaje usunięta, ponieważ użytkownicy zadają bardzo ogólnikowe pytanie, nie umieszczają kodu albo sami nie do końca wiedzą co chcą osiągnąć.

Moderatorzy są odpowiedzialni za odsiewanie pytań wysokiej jakości od tych niskiej jakości. Często zdarza się, że najpierw proszą o uzupełnienie pytania i jeśli pytający nie wprowadzi potrzebnych zmian to dopiero wtedy pytanie zostaje zablokowane i usunięte. Przykładowo w 2018 roku platforma miała ponad 100 milionów aktywnych użytkowników, którzy zadali 2 miliony nowych pytań. Nietrudno wyobrazić sobie, że gdyby nie wsparcie społeczności i moderatorów – wolontariuszy platforma mogłaby już zostać zalana bardzo niskiej jakości pytaniami. Z problem tym ma pomóc zaprojektowany system do oceny przydatności pytania na podstawie jego tytułu, treści, tagów oraz daty utworzenia.

Baza danych

System został zaprojektowany na podstawie bazy danych zawierającej 60 tys. wybranych pytań z lat 2016-2020 zadanych na platformie StackOverflow. Każdemu wpisowi została przyporządkowana jedna z trzech kategorii:

- **HQ** – wysokiej jakości pytania, z wysoką oceną społeczności i bez edycji
- **LQ_EDIT** – niskiej jakości pytania z negatywnym wynikiem i z licznymi poprawkami społeczności. Pomimo zmian pytanie wciąż pozostaje otwarte
- **LQ_CLOSE** – niskiej jakości pytania, które zostały zamknięte przez społeczność bez ani jednej poprawki

Baza danych została podzielona na dwa pliki:

- Treningowy zawierający 45 tys. rekordów
- Testowy zawierający 15 tys. rekordów

Dane dotyczące każdego pytanie mają następujące meta dane:

1. **Id**: unikalny numer pytania
2. **Title**: tytuł pytania
3. **Body**: treść pytania, zawiera formatowanie HTML
4. **Tags**: tagi przypisane do pytania, rozdzielone przecinkiem
5. **CreationDate**: data utworzenia pytania

Analiza danych

Na potrzeby projektu surowe dane zostały w odpowiedni sposób przetworzone. Na ich podstawie stworzone zostały cechy.

Cechy

1. **Liczba słów kluczowych w pierwszych 20% treści pytania.**

   Cecha opisuje ilość słów kluczowych zawartych w pierwszych 20% treści pytania. Powtarzające się słowa kluczowe są dodawane do tej liczby.

   $$C_1 = \sum_{i=0}^{n} S(k_i, T_{20})$$

   gdzie:

   S – suma wystąpień danego słowa kluczowego w podanym tekście

   k_i – kolejne słowo kluczowe

   T_{20} – pierwsze 20% tekstu

2. **Liczba słów kluczowych w pierwszych 50% treści pytania.**

   Cecha opisuje ilość słów kluczowych zawartych w pierwszych 50% treści. Powtarzające się słowa kluczowe są dodawane do tej liczby.

   $$C_2 = \sum_{i=0}^{n} S(k_i, T_{50})$$

   gdzie:

   S – suma wystąpień danego słowa kluczowego w podanym pytaniu

   k_i – kolejne słowo kluczowe

   T_{50} – pierwsze 50% treści

3. **Gęstość słów kluczowych.**

   Cecha opisuje stosunek liczby słów kluczowych do liczby wszystkich słów w treści pytania. Powtarzające się słowa kluczowe są dodawane do liczby słów kluczowych.

   $$C_3 = \frac{Q}{L}$$

   gdzie:

   Q – suma wystąpień wszystkich słów kluczowych w pytaniu

   L – liczba wszystkich słów w pytaniu

4. **Liczba wszystkich słów w treści pytania.**

   Cecha opisuje ilość słów w treści pytania.

   $$C_4 = L$$

   gdzie:

   L – liczba wszystkich słów w pytaniu

5. **Stosunek usuniętych słów do wszystkich słów w treści pytania.**

   Cecha opisuje procent słów usuniętych do wszystkich słów w pytaniu.

   $$C_5 = \frac{Z}{L}$$

   gdzie:

   Z – Liczba słów wyciętych z pytania

   L – liczba wszystkich słów w pytaniu

6. **Gęstość wektora słów.**

   Cecha opisuje stosunek liczby słów kluczowych do liczby słów wektora tekstu - słowa pozostałe po wycięciu. Powtarzające się słowa kluczowe są dodawane do liczby słów kluczowych.

   $$C_6 = \frac{Q}{T}$$

   gdzie:

   Q – suma wystąpień wszystkich słów kluczowych w pytaniu

   T – zbiór wektora tekstu

7. **Liczba paragrafów występujących w tekście.**

   Cecha opisuję liczbę paragrafów.

   $$C_7 = P$$

   gdzie:

   P – liczba paragrafów w treści pytania

8. **Liczba wstawek z kodem w treści pytania.**

   Cecha opisuję liczbę wstawek z kodem w treści pytania.

   $$C_7 = O$$

   gdzie:

   O – Liczba wstawek <code>..</code> w treści pytania

9. **Najbardziej popularny tag.**

   Dla wszystkich artykułów tworzona jest lista tagów i zostaje posortowana według ilości wystąpień we wszystkich artykułach. Dla danego artykułu zostaje zwrócony tag, który w utworzonym rankingu jest najbardziej popularny.

10. **Średnia pozycja wszystkich tagów.**

    Cecha opisuje ilość słów kluczowych zawartych w pierwszych 50% treści. Powtarzające się słowa kluczowe są dodawane do tej liczby.

    $$C_{10} = \frac{\sum_{i=0}^{n} S(k_i, T)}{n}$$

    gdzie:

    S – suma pozycji na liście popularnych tagów dla tagów w podanym pytaniu

    k_i – pozycja na liście popularnych tagów dla kolejnego tagu w pytaniu

    T– tagi przypisane do danego pytania

Słowa kluczowe

- Słowa wybrane zostały na podstawie ich istotności dla zbioru tekstów, która jest obliczana za pomocą formuły:

  $$V_{TF-IDF} = TF \cdot IDF$$

  gdzie:

  TF - częstotliwość słowa w danym tekście

  IDF - odwrotna częstotliwość występowania słowa we wszystkich tekstach

- **TF (ang. Term Frequency)**

  Dzięki częstotliwości występowania danego słowa można określić ważność tego słowa w danym dokumencie.

  $$TF = \frac{S}{L}$$

  gdzie:

  S - liczba wystąpień danego słowa w danym tekście

  L – liczba wszystkich słów tekstu

- **IDF (ang. Inverse Document Frequency)**

  Dzięki odwrotnej częstotliwości występowania danego słowa można określić w ilu artykułach występuje to słowo.

  $$IDF = log\frac{W}{A}$$

  gdzie:

  W - liczba wszystkich tekstów

  A - liczba tekstów w których występuje dane słowo

Wycinanie słów

W projekcie znajdują się dwie listy: lista poprawnych słów oraz lista najczęściej używanych słów. Przy obliczaniu niektórych cech treść pytań poddana została filtracji. W treści pytań pozostawiano jedynie słowa, które znajdowały się na liście poprawnych słów. Z pozostałych słów dodatkowo usuwano słowa, które znajdowały się na liście najczęściej używanych słów.

Normalizacja

Utworzone cechy zostały poddane normalizacji czyli sprowadzeniem ich wartości do przedziału [0,1] wraz z zachowaniem proporcji każdej wartości.

$$X_i = \left[\frac{X_{i_1} - \min(X_{i_1})}{\max(X_{i_1}) - \min(X_{i_1})}, \dots, \frac{X_{i_n} - \min(X_{i_n})}{\max(X_{i_n}) - \min(X_{i_n})}\right]$$

gdzie:

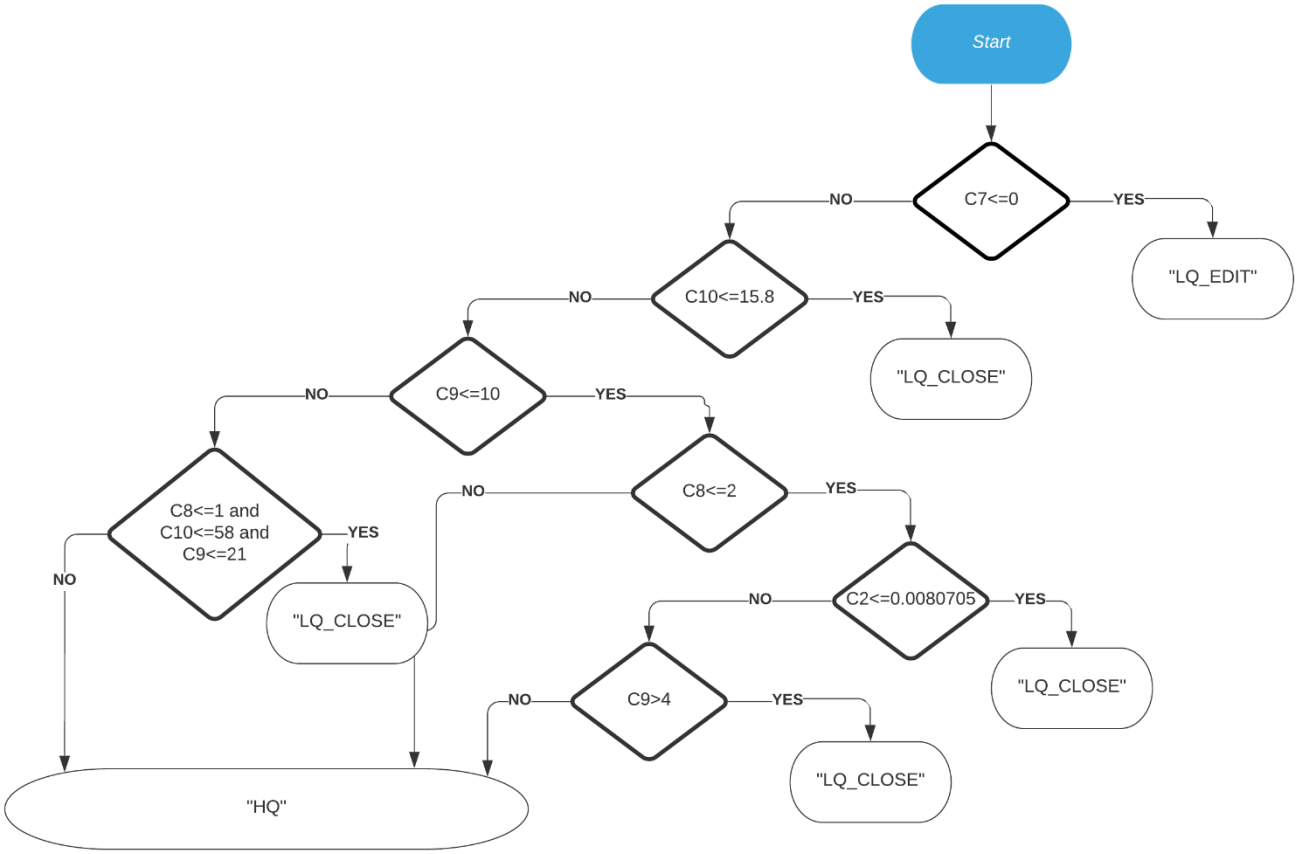
X_i – kolejny wektor cech

$\min(X_i)$ - najmniejsza wartość i-tej cechy z całego zbioru artykułów

$\max(X_i)$ - największa wartość i-tej cechy z całego zbioru artykułów

Reguły

Przetworzone oraz znormalizowane cechy wykorzystano do stworzenia drzewa decyzyjnego. Do stworzenia drzewa decyzyjnego użyto zewnętrznego oprogramowania. Stworzone drzewo przeanalizowano oraz uproszczono. Powstałe w ten sposób reguły umieszczono w projekcie. Sposób ich działania przedstawiono na schemacie:



W stworzonych regułach użyta została tylko jedna reguła wykorzystująca słowa kluczowe (C2 – liczba słów kluczowych w pierwszych 50% tekstu. Manipulowanie poziomem istotności wyszukiwanych słów kluczowych praktycznie nie wpływało na polepszenie wyników. Drzewo decyzyjne natomiast dużo chętniej pod uwagę brało cechy utworzone na podstawie tagów (C9, C10).

System

Technologia

Projekt został przygotowany w języku Python (logika przetwarzania danych) oraz C# (GUI). Wymaga zainstalowania modułu Python 'pythonnet'. Testowane na .NET Framework 4.7.2.

Struktura

Struktura projektu przedstawia się następująco:

- **Data** – folder zawierający projekty
  - **Stack Overflow DB** – zawiera pliki bazy danych CSV oraz reguły jako skrypty Pythona
    - **evaulator.py** – zawiera reguły
    - **sample.csv** – wycinek train.csv
    - **train.csv** – dane trenujące
    - **valid.csv** – dane testowe
- **Docs** – dokumentacja projektu
- **Logs** – Logi zapisywane podczas działania aplikacji
- **main.pyw** – główny skrypt aplikacji
- **MainWindows.xaml** – układ okna
- **utility.py** – przydatne funkcje do użycia w programie

Aplikacja została przygotowana w taki sposób, że można dołączać inne projekty poprzez utworzenie nowego katalogu w folderze Data, a w środku należy umieścić plik (lub pliki) CSV bazy danych oraz plik (lub pliki) klasyfikatora.

Klasyfikator

Klasyfikator to skrypt napisany w jeżyku Python, który zwiera funkcję **examineFile**. Funkcja ta w zależności od argumentu wejściowego isLearning wykonuje jedną z dwóch rzeczy:

isLearning=true

Przygotowuje dane do klasyfikacji poprzez utworzenie nowych kolumn (cech).

dane – słownik zawierający dane w postaci <nazwa kolumny, zawartość> odczytane z pliku CSV, nie zawiera kolumny zawierającej kategorię rekordu.

Funkcja musi zwracać słownik <nazwa, wartość> danych, które mogą zostać użyte do klasyfikacji.

isLearning=false

Klasyfikuje wpis na podstawie jego meta danych.

dane – słownik zawierający dane odczytane z tymczasowego pliku CSV, w którym znajdują się nowo utworzone cechy wygenerowane tą samą funkcją (z argumentem isLearning=true).

Funkcja musi zwracać kategorię, która została rozpoznana na podstawie przekazanych informacji.

Wynik

Na podstawie przeprowadzonych obliczeń program zwraca skuteczność rozpoznania dla wszystkich rekordów oraz dla poszczególnych kategorii.

Instrukcja

Instalacja

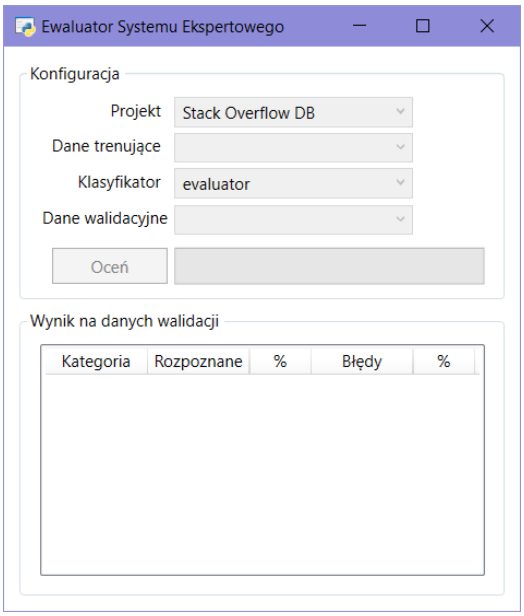
Program nie wymaga instalacji, ale wymaga doinstalowania modułu pythonnet dla Pythona:

```
python –m pip install pythonnet
```

Dodatkowo należy się upewnić, że w systemie zainstalowany został .NET Framework w wersji 4.0+.

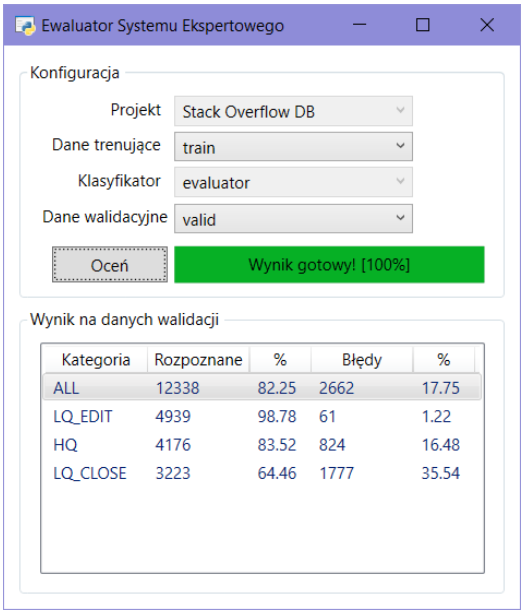
Ocena klasyfikacji

Program uruchamiany poprzez otworzenie skryptu ‘main.py’. Powinno pojawić się następujące okno:



W celu wykonania klasyfikacji należy dokonać kolejno:

1. Wyboru projektu (wybór folderu z katalogu ‘Data’)
2. Wyboru danych trenujących (wybór pliku CSV z folderu projektu, domyślnie zostanie wybrany plik train.csv)
3. Wyboru klasyfikatora (wybór pliku klasyfikatora z folderu projektu)
4. Wyboru danych walidacyjnych (wybór pliku CSV z folderu projektu, domyślnie zostanie wybrany plik valid.csv)
5. Kliknąć na przycisk „Oceń”
6. Podczas obliczeń postęp będzie sygnalizowany poprzez pasek postępu obok przycisku
7. Po zakończeniu klasyfikacji wynik zostanie wyświetlony w dolnej części okna:



Pierwszy wiersz zawsze zawiera podsumowanie „ALL” dla ogólnego wyniku klasyfikacji. Pozostałe wiersze to wyniki rozpoznania poszczególnych kategorii. Tutaj można zauważyć, że klasyfikator nie radzi sobie z rozpoznaniem pytań zamkniętych o niskiej jakości (LQ_CLOSE), ale bardzo dobrze radzi sobie z nie zamkniętymi, niskiej jakości pytaniami LQ_EDIT.

Źródła

- Poprawne słowa angielskie [https://github.com/dwyl/english-words] (dostęp 22.01.2021)
- Lista słów do wykluczenia [https://countwordsfree.com/stopwords] (dostęp 22.01.2021)
- Baz danych [https://www.kaggle.com/imoore/60k-stack-overflow-questions-with-quality-rate] (dostęp 22.01.2021)