Dokumentacja końcowa (wersja wstępna)

Zadanie rozpoznawania tekstu w AMOD

Autorzy:   
Karol Gałązka,  
Tomasz Kogut

# Architektura

Projekt został podzielony na dwa osobne moduły: moduł zajmujący się przygotowaniem danych uczących (DCUpdate) oraz moduł klasyfikacji(BagOfWordsClasifier). Integralną częścią rozwiązania jest wewnętrzna baza danych, która łączy obie części.

# Szczegółowy opis budowy klasyfikatora

BagOfWordsTextClassifier

Udostępnia interfejs:

* int[] ProcedureRecognition(string text)
* int[] NextStagePrediciton(int procedurId, int phaseId, string text)
* int[] NextPersonPrediction(int procedurId, int phaseId, string text)

Poprzez te metody można dokonać kolejno klasyfikacji: tekstu do procedury, następnego etapu procedur na podstawie tekstu, oraz osoby do której należy przekazać sprawę.

Klasyfikator opiera się na typach danych opisanych w punkcie **Error! Reference source not found.**. Na ich podstawie przed pierwszym użyciem tworzony jest szereg macierzy i danych pomocniczych, które są wykorzystywane do szybkich obliczeń.

* private DBRepresentation allWords = null;
* private AllCases CasesSet = null;
* private AllProcedures ProceduresSet = null;
* private AllDecisionsPhase PhaseDecisionData = null;
* private AllDecisionsPeople PeopleDecisionData = null;

Dla zmniejszenia złożoności obliczeniowej ograniczamy rozmiar wektora słów, który będzie reprezentował sprawę. Tworzona jest mapa, która słowu przypisuje miejsce w wektorze słów - FetchMeaningfulWords. Brane są pod uwagę tylko te słowa, które występują mniej niż w MaximumFrequency procent spraw. Na podstawie tego współczynnika i ilości słów obliczany jest próg maksymalnej dopuszczalnej liczby wystąpień słowa powyżej, którego słowo jest odrzucane.

* Dictionary<String, int> MapWordToColumn
* const double MaximumFrequency = 0.9;
* int wordThreshold = int.MaxValue;
* void FetchMeaningfulWords()

Ze zbioru procedur wczytywane są wagi słów, które zostały uznane za ważne i umiejscawiane się w macierzy ProcedureMatrix. Wiersze tej macierzy odpowiadają procedurom, kolumny słowom. Kolumny zgadzają się z tymi zapisanymi w MapWordToColumn. Dodatkowo tworzona jest mapa, która odwzorowuje wiersze macierzy na identyfikatory procedur w bazie danych – MapRowToProcedureId, tak by po uzyskaniu najlepszego dopasowania, można było zidentyfikować procedurę.

* double[][] ProcedureMatrix
* int[] MapRowToProcedureId

Ponieważ obliczenia cosinusów odywają się na typie podstawowym double powinny być one szybkie. Do ich wykonywania używany jest stworzona klasa VectorOperations, która operuje na tablicach liczb zmiennoprzecinkowych podwójnej precyzji.

* class VectorOperations
* static double VectorOperations.VectorsConsine(double[] v1, double[] v2)

# Szczegółowy opis modułu dostarczającego danych uczących

Dostęp do bazy danych AMOD realizowany jest przez moduł DCUpdate. Monitoruje on bazę danych AMOD i w przypadku pojawienia się nowych rekordów aktualizuje odpowiednie dane uczące. Służy do tego funkcja

* public void update()

klasy AmodDBTools, która przetwarza wszelkie nowe dane, dodane od ostatniego procesu akutalizacji.

Mamy do czynienia z następującymi sytuacjami: dodaniem oraz modyfikacją danych.

## Dodanie nowych danych

Dodanie nowych danych zostało rozbite na dwa przypadki: dodanie nowej sprawy oraz dodanie nowego załącznika.

## 1.Dodanie nowej sprawy

Dodanie nowego dokumentu wymaga ponownego przeliczenia wszystkich współczynników TF-IDF dla danych uczących, gdyż zmianie ulega wartość log(|D|) (D – liczba dokumentów)

## 3.2.2.Dodanie nowego załącznika

Dodanie nowego załącznika zostało wydzielone ze względu na mniejszy wymagany nakład pracy. Dla słów, które znajdowały się już w danym casie wystarczy jedynie zwiększyć wartość współczynnika TF-IDF o wielokrotność ich wystąpienia. Dla nowych słów należy wykonać obliczenia dla wszystkich ich wystąpień również w innych sprawach, gdyż uległa zmianie wartość log(DF).

## Modyfikacja danych

Na obecnym etapie nie została zaimplementowana ta funkcjonalność

## Obliczanie współczynnika TF-IDF

Aby przyspieszyć wykonanie obliczeń współczynnika TF-IDF została zaimplementowana struktura IDFCalculation, która przechowuje wartości IDF oraz log(DF(w)) dla wszystkich słów znajdujących się w systemie oraz dotarcza metod pozwalających je modyfikować. Wzór na IDF podany w treści zadania został przekształcony na różnicę algorytmów:

IDF(w) = log(|D|) – log( DF(w)),

Log(|D|) obliczane jest przy każdej zmianie liczby dokumentów znajdujących w bazie i nie musi być ponownie wyliczane.

Do obliczeń współczynnika TF-IDF konieczne było również przechowywanie wartości TF. W tym celu powstała struktura CasesTF

# Komunikacja pomiędzy modułami – opis wewnętrznej bazy danych

Wewnętrzna baza danych przechowuje dwa rodzaje danych:

* Dane wspomagające działanie modułu dostarczającego dane uczące (wspomniane powyżej CasesTF oraz IDFCalculation),
* Hashmapy z danymi uczącymi.

Hashmapy z danymi uczącymi to:

* AllCases – reprezentacja spraw
* DBRepresentation – reprezentacja decyzji
* AllProcedures – reprezentacja procedur
* AllDecisionsPeople – reprezentacja decyzji (następnej osoby)
* AllDecisionPhase - reprezentacja decyzji (następnego etapu)

Przy stacie moduł dostarczający dane uczące pobiera obiekty z wewnętrznej bazy, a następnie cyklicznie będzie dostarczał nowe wersje.

Moduł podejmowanai decyzji używa jedynie hashmap z danymi uczącymi. W przypadku pojawienia się nowych wersji, cyklicznie pobiera zakutalizowane obiekty z bazy.