Informatyka, studia dzienne, I st. semestr VI

Komputerowe systemy rozpoznawania **2018/2019**
Prowadzący: dr inż. Marcin Kacprowicz poniedziałek, 14:10

Data oddania: _____ Ocena: _____

Justyna Hubert 210200
Karol Podlewski 210294

Zadanie 1: Ekstrakcja cech, miary podobieństwa, klasyfikacja*

1. Cel

Celem zadania było stworzenie aplikacji do klasyfikacji tekstów metodą k-NN, korzystając z różnych sposób ekstrakcji wektorów cech oraz istniejących miar podobieństwa porównać kategorie do tych przypisanych przez aplikację.

2. Wprowadzenie

Zagadnieniem, jakim zajmowaliśmy się w ramach projektu jest klasyfikacja statystyczna, która jest rodzajem algorytmu statystycznego przydzielającego elementy do klas, bazując na cechach tych elementów. W ramach przeprowadzanego eksperymentu zaimplementowaliśmy klasyfikator k-najbliższych sąsiadów.

Algorytm k najbliższych sąsiadów, nazywany także algorytmem k-NN, należy do grupy algorytmów leniwych, czyli takich, które nie tworzą wewnętrznej reprezentacji danych uczących, lecz szukają rozwiązania dopiero w momencie pojawienia się wzorca testującego. Przechowuje wszystkie wzorce uczące, względem których wyznacza odległość wzorca testowego [2]. Metoda

* SVN: `https://github.com/hubjust/KSR`

k-NN wyznacza k sąsiadów, do których badany element ma najmniejszą odległość w danej metryce, a następnie wyznacza wynik w oparciu o najczęstszy element, wśród k najbliższych. W przypadku naszego projektu odległość definiujemy jako skalę podobieństwa tekstów.

W ramach zadania zostały użyte 2 metody ekstrakcji cech:

— Term frequency - metoda polegająca na zliczeniu częstości występowania danego słowa w dokumencie. Obliczana jest z poniższego wzoru:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k nk, j}$$

— Inverse document frequency - metoda polegająca na wyznaczeniu, czy dane słowo występuje powszechnie we wszystkich dokumentach. Jest to logarytmicznie skalowana odwrotna część dokumentów zawierających wybrane słowo (uzyskana poprzez podzielenie całkowitej liczby dokumentów przez liczbę dokumentów zawierających ten termin). Obliczana jest z poniższego wzoru:

$$idf_i = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

Do obliczenia odległości tekstów posłużyliśmy się 3 metrykami:

— metryka Euklidesowa - w celu obliczenia odległości $d_e(x, y)$ między dwoma punktami x, y należy obliczyć pierwiastek kwadratowy z sumy drugich potęg różnic wartości współrzędnych o tych samych indeksach, zgodnie ze wzorem:

$$d_e(x, y) = \sqrt{(y_1 - x_1)^2 + \cdots + (y_n - x_n)^2}$$

— metryka uliczna (Manhattan, miejska) - w celu obliczenia odległości $d_e(x, y)$ między dwoma punktami x, y należy obliczyć sumę wartości bezwzględnych różnic współrzędnych punktów x oraz y, zgodnie ze wzorem:

$$d_m(x, y) = \sum_{k=1}^{n} |x_k - y_k|$$

— metryka Czebyszewa - w celu obliczenia odległości $d_e(x, y)$ między dwoma punktami x, y należy obliczyć maksymalną wartość bezwzględnych różnic współrzędnych punktów x oraz y, zgodnie ze wzorem:

$$d_{ch}(x, y) = \max_i |x_i - y_i|$$

3. Opis implementacji

Program został stworzony w języku C#. Graficzny interfejs użytkownika został stworzony przy wykorzystaniu Windows Presentation Foundation. Logika aplikacji została odseparowana od GUI, w zgodzie ze wzorcem projektowym Model-view-viewmodel (MVVM), poprzez implementacje trzech projektów (Logic, ViewModel i GUI).

3.1. Logic

Klasy Chebyshev, Euclidean oraz Manhattan odpowiadają za prawidłowe obliczenia odległości tekstów. Dziedziczą one z klasy abstrakcyjnej Metric.

Klasa Article odwzorowuje artykuły wczytane do programu. Przechowuje informacje o dokumencie takie jak: tytuł, tekst, kategorie, przypisane etykiety, wektor cech, odległość.

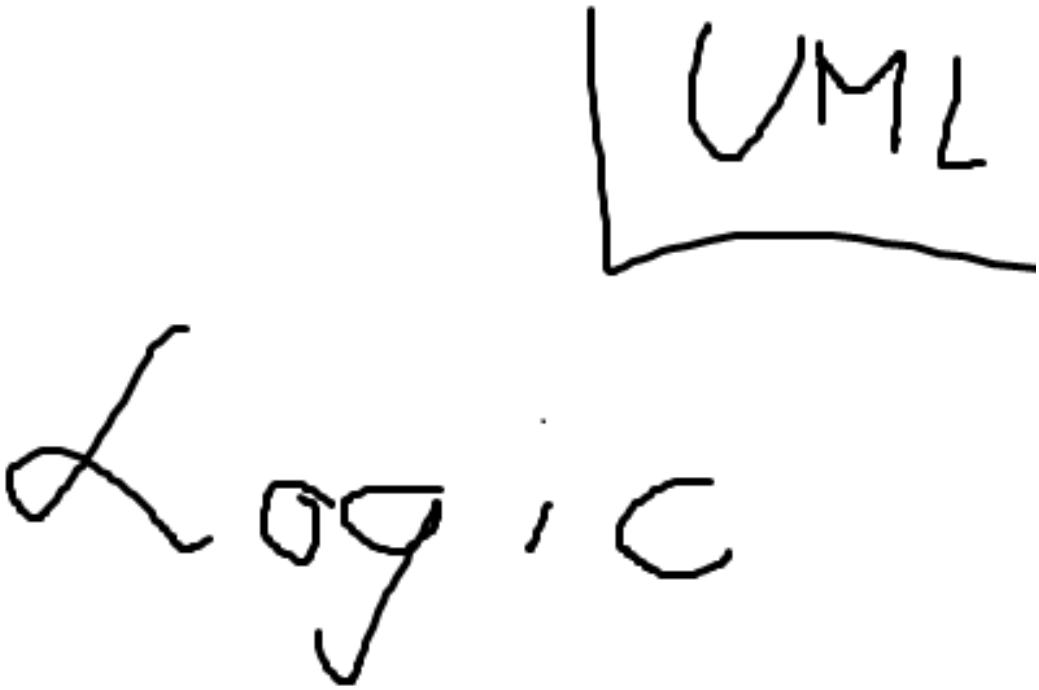
Klasa FeatureExtraction implementuje dwie metody ekstrakcji cech - term frequency oraz inverse document frequency.

Klasa FileReader odpowiada za poprawne wczytywanie plików do programu - wyselekcjonowanie wybranych przez nas informacji (tytuł, teksy, przypisane etykiety) i na ich podstawie stworzenie obiektu klasy Article. Z wczytanego tekstu usuwane są słowa, które występują w wykorzystanej przez nas stop liście [3]. Ten zabieg ma za zadanie wykluczyć terminy, które nie wnoszą kluczowych, dla nas, informacji. Następnie, tekst zostaje poddane stemizacji, czyli usunięciu ze słowa końcówki fleksyjnej pozostawiając tylko rdzeń wyrazu. Do tego także wykorzystujemy zewnętrzną bibliotekę [4].

Klasa KnnAlgorithm odpowiada za implementację algorytmu k-najbliższych sąsiadów. W tym miejscu wyliczane są wystąpienia słów w podanych dokumentach.

Klasa Sets odpowiedzialna jest za podział wczytanych artykułów na dane testowe oraz treningowe.

Klasa CategoryCompatibilityChecker ma za zadanie sprawdzić, czy dany artykuł zawiera jedną etykietę z danej kategorii, oraz czy ta etykieta jest brana pod uwagę przy analizowaniu danej kategorii.

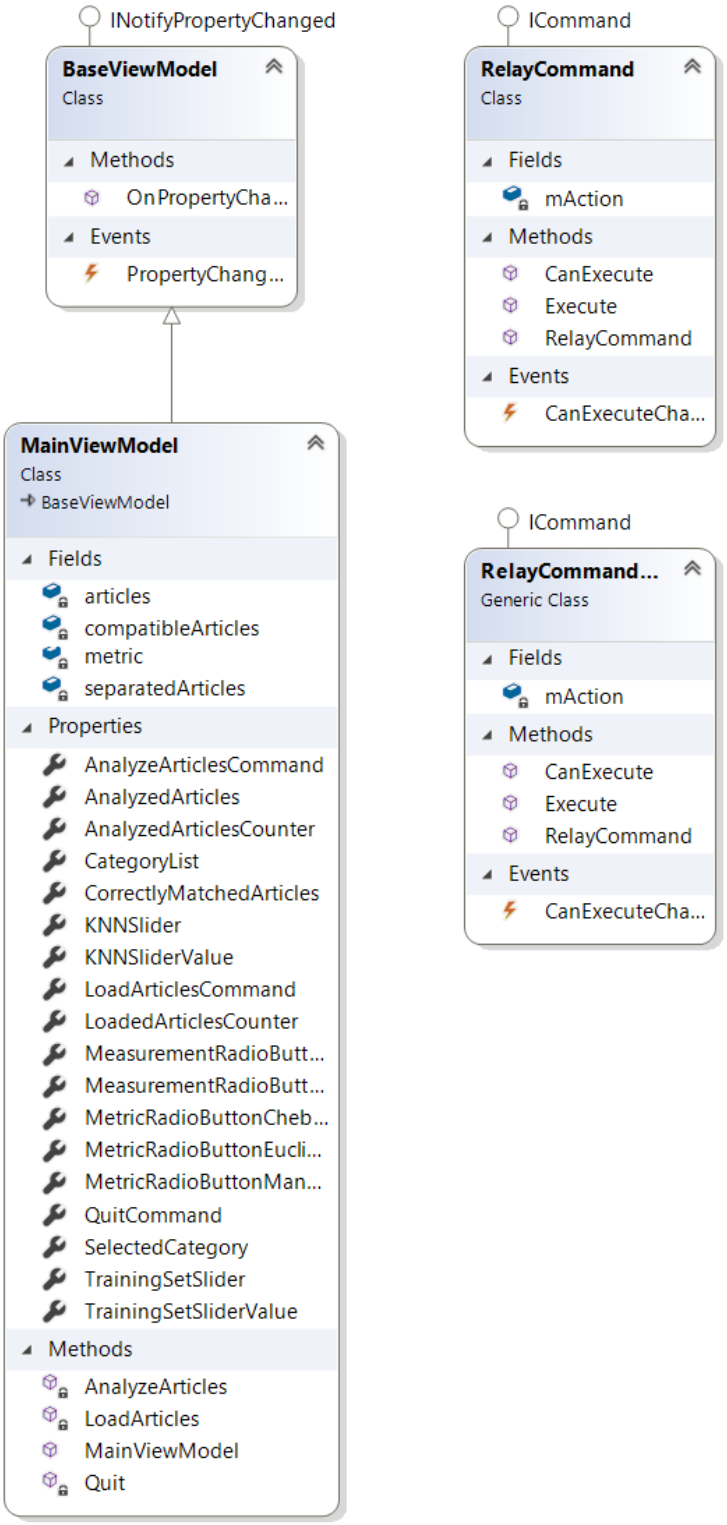


Rysunek 1. Diagram UML wygenerowany dla projektu Logic

3.2. ViewModel

Projekt ViewModel ma za zadanie odseparować logikę programu od interfejsu graficznego.

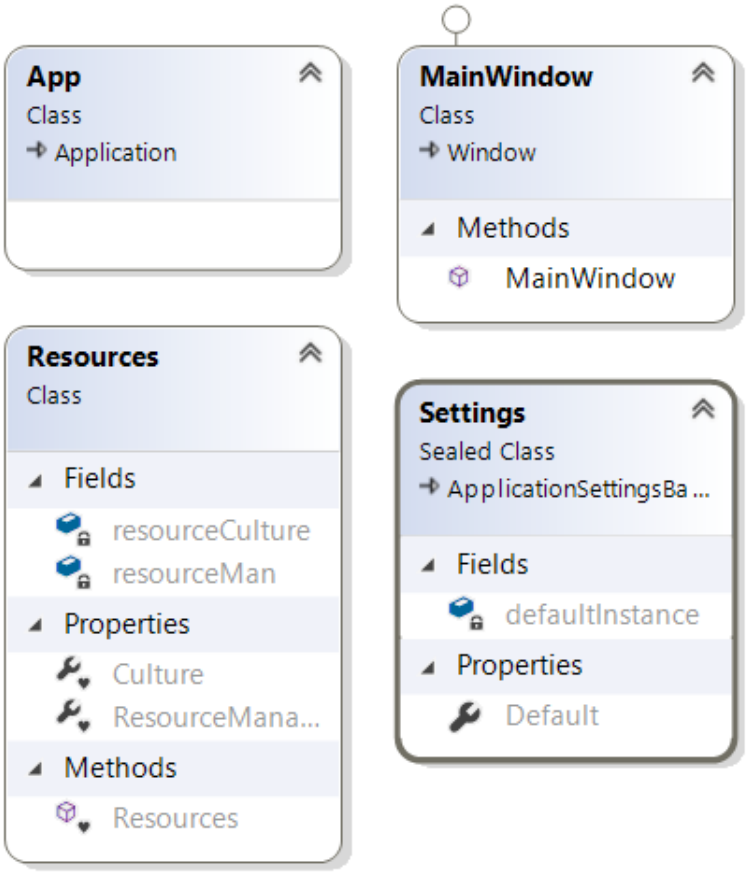
Klasa MainViewModel przyjmuje dane wejściowe od użytkownika i reaguje na jego poczynania wywołując wybrane akcje z logiki programu oraz odpowiada za odświeżanie widoków w interfejsie graficznym.



Rysunek 2. Diagram UML wygenerowany dla projektu MainViewModel

3.3. GUI

Projekt GUI (graphical user interface) implementuje przejrzysty oraz łatwy w obsłudze graficzny interfejs użytkownika.



Rysunek 3. Diagram UML wygenerowany dla projektu GUI

4. Materiały i metody

Klasyfikacja tekstów została wykonana wszystkimi dostępnymi metodami ekstrakcji cech dla wszystkich trzech metryk. Dla każdego przypadku testowego dokonano klasyfikacji tekstu dla k $\in \{2, 3, 5, 7, 10, 15, 20\}$ najbliższych sąsiadów. Wyniki porównano z faktyczną etykietą danego artykułu. Za każdym razem zbiór treningowy stanowił 60% artykułów, zaś zbiór testowy 40%.

Klasyfikacja dotycząca lokalizacji przeprowadzana była jedynie na danych, których pole places przyjmowało jedną z wartości: west-germany, usa, france, uk, canada, japan.

Klasyfikacja dotycząca tematów przeprowadzana była jedynie na danych, które pole topics przyjmowało jedną z wartości: gold, cocoa, sugar, coffe, grain.

Klasyfikacja własnych tesktów przeprowadzana była na danych, których

pole author przyjmowało jedną z wartości: taylor swift, macklemore, twenty one pilots, eminem, ed sheeran, black eyed peas.

5. Wyniki

Wyniki dla kolejnych eksperymentów przedstawiają tabele. Tabele 1-3 przedstawiają odpowiednio eksperymenty przeprowadzone dla metryki Euklidesa, ulicznej oraz Czebyszewa przy użyciu Term frequency, natomiast Tabele 4-6 przedstawiają alogiczne dane dla Inverse document frequency. Zamieszczone wykresy mają za zadanie porównać jak dobrze radzą sobie metryki z daną kategorią.

k	**places** [%]	**topics** [%]	**authors** [%]
2	74.4	53.7	39.0
3	78.5	52.2	43,9
5	80.2	52.2	48,8
7	81.0	53.7	36,6
10	81.5	60.4	36,6
15	81.6	62.7	39,0
20	81.4	61.2	36,6

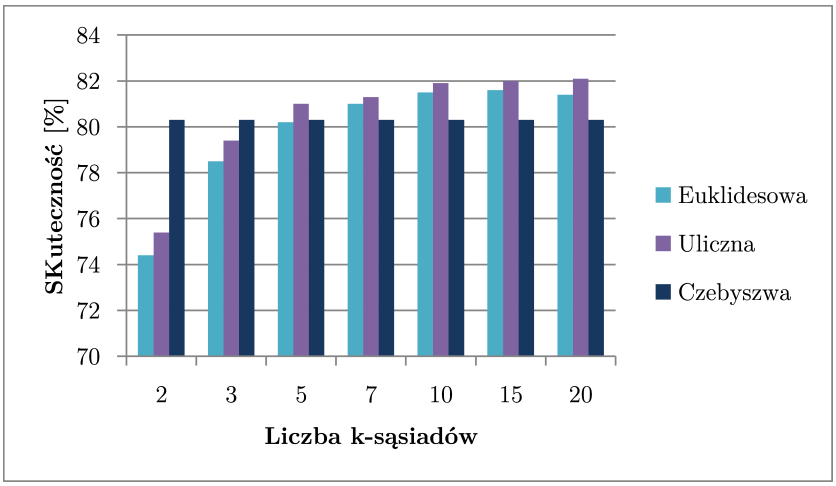
Tabela 1. Skuteczność klasyfikacji dla metryki Euklidesowej dla TF

k	**places** [%]	**topics** [%]	**authors** [%]
2	75.4	56.7	36,6
3	79.4	56.7	41,5
5	81.0	61.2	43,9
7	81.3	59.0	36,6
10	81.9	64.9	41,5
15	82.0	64.9	39,0
20	82.1	63.4	36,6

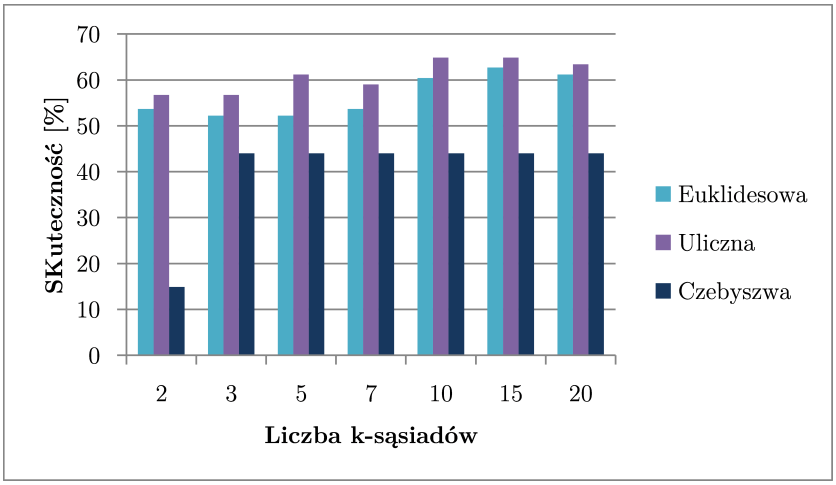
Tabela 2. Skuteczność klasyfikacji dla metryki ulicznej dla TF

k	**places** [%]	**topics** [%]	**authors** [%]
2	80.3	14.9	17.1
3	80.3	44.0	17.1
5	80.3	44.0	17.1
7	80.3	44.0	17.1
10	80.3	44.0	17.1
15	80.3	44.0	17.1
20	80.3	44.0	17.1

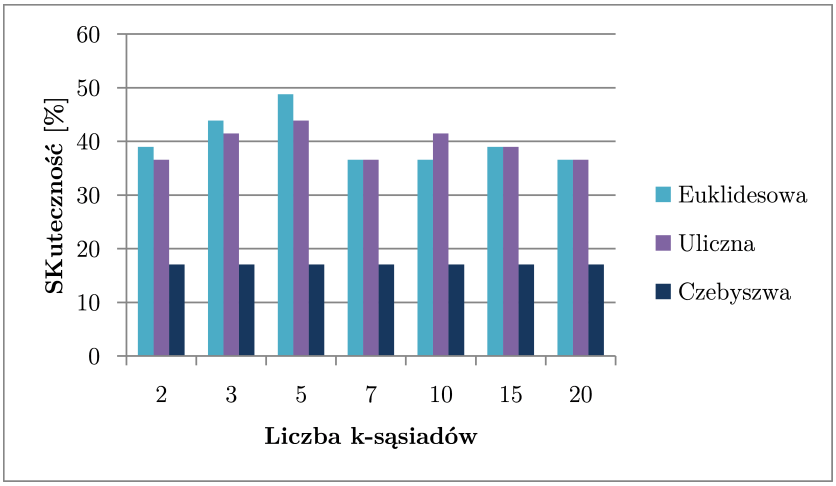
Tabela 3. Skuteczność klasyfikacji dla metryki Czebyszewa dla TF



Rysunek 4. Dane z Tabel 1-3 dla kategorii places



Rysunek 5. Dane z Tabel 1-3 dla kategorii topics



Rysunek 6. Dane z Tabel 1-3 dla kategorii authors (własne teksty)

k	places [%]	topics [%]	authors [%]
2	79,0	63,4	19,5
3	82,0	64,2	17,1
5	82,1	59,0	29,3
7	83,3	62,1	19,5
10	82,0	64,9	22,0
15	81,9	67,9	14,6
20	81.1	67,1	14,6

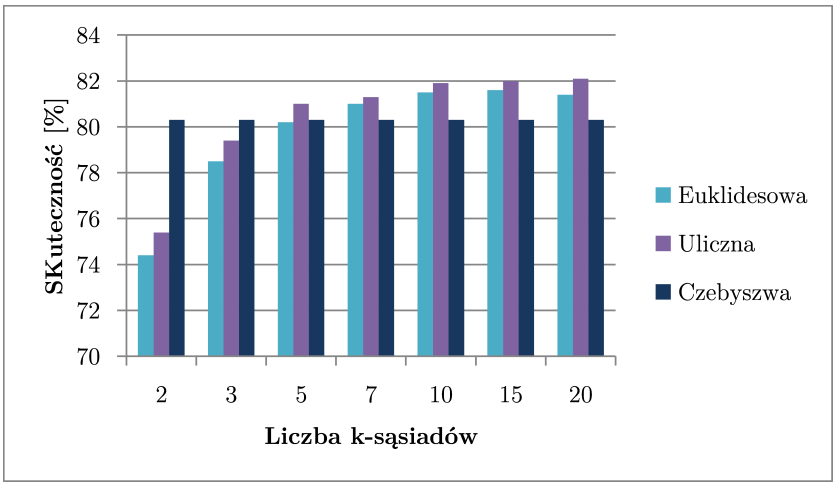
Tabela 4. Skuteczność klasyfikacji dla metryki Euklidesowej dla IDF

k	places [%]	topics [%]	authors [%]
2	80,2	59,7	19,5
3	82,4	65,7	17,1
5	82,6	67,2	29,3
7	83,3	67,2	16,5
10	82,6	67,2	22.0
15	82,1	67,2	14,6
20	81,6	67,9	14,6

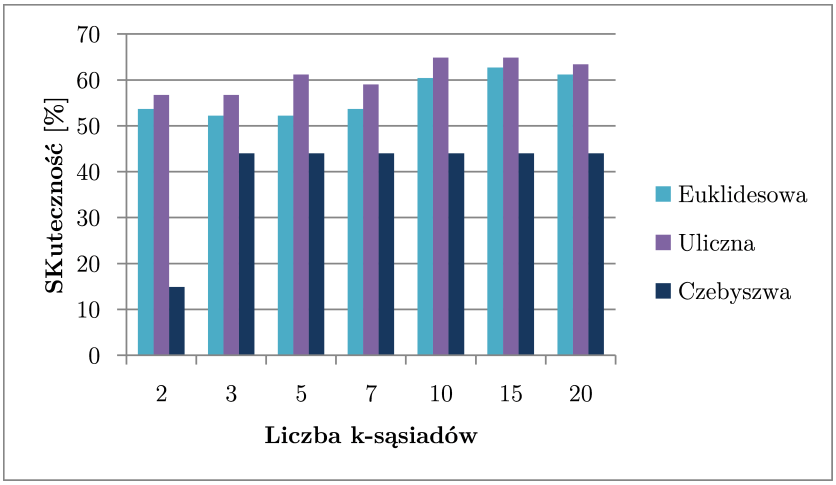
Tabela 5. Skuteczność klasyfikacji dla metryki ulicznej dla IDF

k	places [%]	topics [%]	authors [%]
2	77.0	14.9	17.1
3	77.0	44.0	17.1
5	77.0	44.0	17.1
7	77.0	44.0	17.1
10	77.0	44.0	17.1
15	77.0	44.0	17.1
20	77.0	44.0	17.1

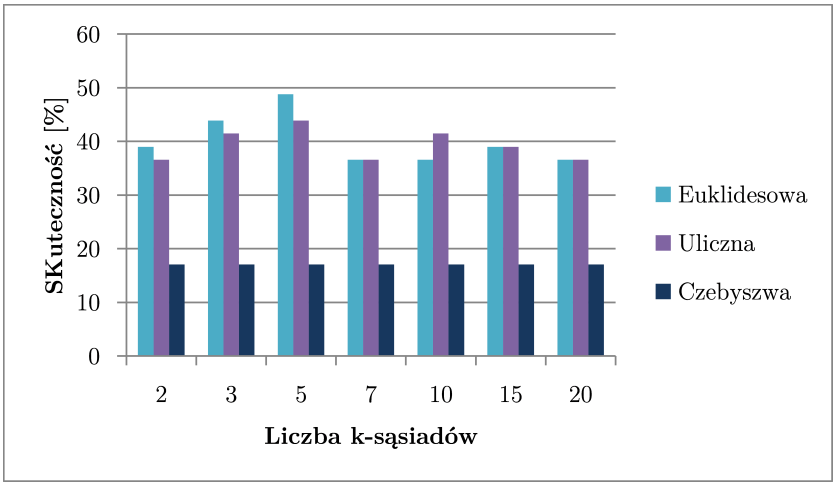
Tabela 6. Skuteczność klasyfikacji dla metryki Czebyszewa dla IDF



Rysunek 7. Dane z Tabel 4-6 dla kategorii places



Rysunek 8. Dane z Tabel 4-6 dla kategorii topics



Rysunek 9. Dane z Tabel 4-6 dla kategorii authors (własne teksty)

6. Dyskusja

Po przeprowadzonych badaniach jesteśmy w stanie zauważyć, iż metryka uliczna charakteryzuje się największą skutecznością w przypadku pierwszego, jak i drugiego sposobu ekstrakcji cech. Niewiele mniejszą skuteczność wykazuje metryka Euklidesowa. Z kolei, najmniejszą skuteczność reprezentuje metryka Czebyszewa. W przeciwieństwo do dwóch pozostałych metryk, zmiana ilości k sąsiadów nie wpływała lub wpływała nieznacząco na liczbę dopasowanych dokumentów.
Niska liczba liczba k-sąsiadów (od 2 do 5) zazwyczaj oraz bardzo wysoka (20) zazwyczaj negatywnie wpływały na skuteczność. Optymalna liczba sąsiadów do dokonania klasyfikacji w przypadku algorytmu k-nn wynosi 10 w przypadku ekstrakcji metodą term frequency oraz 7 w przypadku ekstrakcji inverse document frequency.
Dwie użyte przez nas metody ekstrakcji cech wykazują bardzo podobną skuteczność - inverse document frequency nieznacznie przewyższa term frequency. Może to wynikać z faktu, iż IDF przeszukuje wszystkie dokumenty, a TF tylko jeden.

7. Wnioski

Literatura

[1] Methods for the linguistic summarization of data - aplications of fuzzy sets and their extensions, Adam Niewiadomski, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2008
[2] http://home.agh.edu.pl/ horzyk/lectures/miw/KNN.pdf
[3] https://github.com/hklemp/dotnet-stop-words
[4] http://snowball.tartarus.org/algorithms/english/stemmer.html