Informatyka, studia dzienne, I st. semestr VI

Komputerowe systemy rozpoznawania **2019/2020**
Prowadzący: dr inż. Marcin Kacprowicz poniedziałek, 12:00

Data oddania: _____ Ocena: _____

Radosław Grela 216769
Jakub Wąchała 216914

Zadanie 1: ekstrakcja cech, miary podobieństwa, klasyfikacja*

1. Cel

Celem naszego zadania było stworzenie aplikacji do klasyfikacji tekstów za pomocą metody k-NN (k najbliższych sąsiadów) oraz różnych metryk i miar podobieństwa, a następnie porównać kategorie z tymi wygenerowanymi przez aplikację.

2. Wprowadzenie

Głównym zagadnieniem projektowym, z którym mieliśmy do czynienia w ramach zadania 1 była klasyfikacja statystyczna tekstów na podstawie wektora wyekstrahowanych cech. Do przeprowadzenia eksperymentu zaimplementowaliśmy algorytm *k-najbliższych sąsiadów*.

Algorytm k-najbliższych sąsiadów *(k-NN - k-nearest neighbors)* to jeden z algorytmów zaliczanych do grupy algorytmów leniwych. Jest to taka grupa algorytmów, która szuka rozwiązania dopiero, gdy pojawia się wzorzec testujący. Przechowuje wzorce uczące, a dopiero później wyznacza się odległość wzorca testowego względem wzorców treningowych. [8]

Algorytm ten działa w taki sposób, że dla każdego wzorca testowego obliczana jest odległość za pomocą wybranej metryki względem wzorców treningowych, a następnie wybierana jest k najbliższych wzorców treningowych.

* Github: `https://github.com/Bonniu/KSR`

Wynik wyznaczony jest jako najczęstszy element wśród nich. W naszym zadaniu odległość ta jest równa skali podobieństwa tekstów, a im ta odległość jest mniejsza, tym lepiej.

2.1. Ekstrakcja cech

Do ekstrakcji cech charakterystycznych tekstu utworzyliśmy wektor cech, który opisuje tekst za pomocą 11 cech. Liczba słów zawsze jest liczona po zastosowaniu stop-listy oraz stemizacji, bez znaków przestankowych.

- C_1 - Stosunek słów kluczowych do wszystkich słów w pierwszych 10% tekstu. Obliczona jest za pomocą wzoru:

$$C_1 = s_{k10}/s_{10} \tag{1}$$

  gdzie
  s_{k10} - liczba słów kluczowych,
  s_{10} - liczba wszystkich słów w pierwszych 10% tekstu.
  Przed normalizacją cecha C_1 zawierała się w wartościach $\in [0, 1]$.
- C_2 - Stosunek słów kluczowych do wszystkich słów w ostatnich 10% tekstu. Obliczona jest za pomocą wzoru:

$$C_2 = s_{k90}/s_{90} \tag{2}$$

  gdzie
  s_{k90} - liczba słów kluczowych,
  s_{90} - liczba wszystkich słów w ostatnich 10% tekstu.
  Przed normalizacją cecha C_2 zawierała się w wartościach $\in [0, 0.5]$.
- C_3 - Stosunek słów kluczowych do wszystkich słów w dokumencie. Obliczona jest za pomocą wzoru:

$$C_3 = s_k/s \tag{3}$$

  gdzie
  s_k - liczba słów kluczowych,
  s - liczba wszystkich słów w dokumencie.
  Przed normalizacją cecha C_3 zawierała się w wartościach $\in [0, 0.155]$.
- C_4 - Stosunek słów kluczowych, których ilość liter $\in$ (0,4] do wszystkich słów w dokumencie. Obliczona jest za pomocą wzoru:

$$C_4 = s_k/s \tag{4}$$

  gdzie
  s_k - liczba słów kluczowych, których ilość liter $\in$ (0,4],
  s - liczba wszystkich słów w dokumencie.
  Przed normalizacją cecha C_4 zawierała się w wartościach $\in [0, 0.075]$.
- C_5 - Stosunek słów kluczowych, których ilość liter jest $\geq$8 do wszystkich słów w dokumencie. Obliczona jest za pomocą wzoru:

$$C_5 = s_k/s \tag{5}$$

  gdzie
  s_k - liczba słów kluczowych,
  s - liczba wszystkich słów w dokumencie.
  Przed normalizacją cecha C_5 zawierała się w wartościach $\in [0, 0.1]$.

- C_6 - Stosunek linii do ilości akapitów. Obliczona jest za pomocą wzoru:

$$C_6 = l/a \tag{6}$$

  gdzie
  l - liczba linii,
  a - liczba akapitów.
  Przed normalizacją cecha C_6 zawierała się w wartościach $\in [1, 14]$.
- C_7 - Stosunek słów, których ilość liter jest większa niż 6 do wszystkich słów. Obliczona jest za pomocą wzoru:

$$C_7 = s_6/s \tag{7}$$

  gdzie
  s_6 - liczba słów których ilość liter jest większa niż 6,
  s - liczba wszystkich słów w dokumencie.
  Przed normalizacją cecha C_7 zawierała się w wartościach $\in [0, 0.591]$.
- C_8 - Stosunek słów kluczowych, których ilość liter jest $\leq$6 do wszystkich słów w dokumencie. Obliczona jest za pomocą wzoru:

$$C_8 = s_{6m}/s \tag{8}$$

  gdzie
  s_{6m} - liczba słów kluczowych, których ilość liter jest $\leq$6,
  s - liczba wszystkich słów w dokumencie.
  Przed normalizacją cecha C_8 zawierała się w wartościach $\in [0.409, 1]$.
- C_9 - Ilość słów unikalnych. Jest to liczba słów, które wystąpiły w tekście co najmniej raz. Przykładowo, dla zdania *„Być albo nie być"* ilość słów unikalnych jest równa 3 (*być, albo, nie*).
  Przed normalizacją cecha C_9 przyjmuje wartości $\in [1, 420]$.
- C_{10} - Ilość słów, których ilość liter $\in$ [5,8]. Pseudokod obliczający wartość cechy C_{10}:
  — *C_{10}=0*
  — *Dla każdego słowa w artykule:*
    — *Jeżeli długość słowa>=5 i długość słowa <=8:*
      — *C_{10}++;*
  — *Zwróć C_{10}*
  Przed normalizacją cecha C_{10} zawierała się w wartościach $\in [1, 574]$.
- C_{11} - Najczęściej występujące słowo kluczowe. Jest to cecha tekstowa, której podobieństwo z innym słowem mierzymy jedną z dwóch miar podobieństwa ciągów znaków opisanych w sekcji *Metryki i miary podobieństwa*.

2.2. Wyznaczanie słów kluczowych

Wyznaczenie słów kluczowych przebiega w następujący sposób: na początek za pomocą klasy WordCounter zliczane są wszystkie słowa w artykułach oraz jednocześnie dodawane do odpowiednich list w tej klasie. Każda zmienna jest listą stringów o nazwie wordCountDictionary + nazwa kraju. Dodatkowo, przechowywany jest słownik typu <*string, int*>, którego kluczem jest

słowo, a wartość to ilość wystąpień tego słowa we wszystkich artykułach. Po podliczeniu wszystkich słów oraz przydzieleniu do odpowiednich list wybieramy po 18 najpopularniejszych słów dla każdego kraju, które występują tylko w tym jednym konkretnym kraju. Na koniec $18 * 6 = 108$ słów zostaje słowami kluczowymi. Cały proces wyznaczania słów kluczoywch jest dokonywany po zastosowaniu stop-listy oraz po stemizacji. Ponadto, proces wybierania słów kluczowych pomija 20% wszystkich podliczonych słów, aby proces dopasowywania słów kluczowych do krajów nie trwał zbyt długo.

2.3. Metryki i miary podobieństwa

Do liczenia odległości pomiędzy artykułami oraz obliczenia miary podobieństwa używaliśmy 3 metryk i 2 miar podobieństwa ciągów tekstowych.

1. Metryka Euklidesowa - aby obliczyć odległość $d_e(x, y)$ między wektorami x i y należy obliczyć pierwiastek kwadratowy z sumy kwadratów różnic wartości współrzędnych wektora o tych samych indeksach. Wzór jest następujący [5]:

$$d_e(x, y) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2} \tag{9}$$

   gdzie x_i i y_i to cechy wektora.
2. Metryka uliczna - odległość $d_m(x, y)$ jest równa sumie wartości bezwzględnych z różnic wartości współrzędnych wektora o tych samych indeksach [3]:

$$d_m(x, y) = \sum_{n=1}^{N} |x_n - y_n| \tag{10}$$

   gdzie x_i i y_i to cechy wektora.
3. Metryka Czebyszewa - odległość $d_c(x, y)$ w tej metryce jest równa maksymalnej wartości bezwzględnych różnic współrzędnych punktów x oraz y, zgodnie ze wzorem [4]:

$$d_c(x, y) = \max_i |x_i - y_i| \tag{11}$$

   gdzie x_i i y_i to cechy wektora.
4. Miara *n*-gramów - metoda ta określa podobieństwo łańcuchów tekstowych s_1, s_2 w oparciu o ilość wspólnych podciągów n-elementowych, czyli *n*-gramów [2]:

$$sim_n(s_1, s_2) = \frac{1}{N - n + 1} \sum_{i=1}^{N-n+1} h(i) \tag{12}$$

   gdzie
   $h(i) = 1$, jeśli n-elementowy podciąg zaczynający się od i-tej pozycji w s_1 występuje co najmniej raz w s_2, w przeciwnym razie $h(i) = 0$
   $N - n + 1$ - ilość możliwych n-elemenetowych podciągów w s_1.
   W naszym programie n jest stałe i wynosi 3.

5. Uogólniona miara *n-gramów* (Miara Niewiadomskiego) - ta miara jest ulepszoną wersją miary n-gramów. Bada ona podobieństwo poprzez sprawdzenie podciągów różnej długości od jedno- do N-elementowych, gdzie N jest długością słowa [2]:

$$\mu_N(s_1, s_2) = \frac{2}{N^2 + N} \sum_{i=1}^{N(s_1)} \sum_{j=1}^{N(s_1)-i+1} h(i, j) \tag{13}$$

gdzie
$h(i, j) = 1$, jeśli i-elementowy podciąg w słowie s_1 zaczynający się od j-tej pozycji w słowie s_1 pojawia się co najmniej raz w słowie s_2, w przeciwnym razie $h(i, j) = 0$
$N(s_1), N(s_2)$ – ilość liter w słowach s_1 i s_2;
$N = max\{(s_1), N(s_2)\}$
$\frac{N^2+N}{2}$ - ilość możliwych podciągów od 1-elementowych do N-elementowych w słowie o długości N.

Aby porównać wektory za pomocą metryk w algorytmie k-NN, najpierw wyznaczamy miarę podobieństwa z ostatniej, 11 cechy, która jest cechą tekstową. Wyznaczamy ją za pomocą jednej z dwóch miar. Ponieważ w tych miarach im bliżej 1, tym lepiej, odejmujemy tą liczbę od 1, a następnie używamy jej w metryce.

2.4. Miary jakości

W wynikach klasyfikacji używamy następujących miar jakości [9]:

- Accuracy:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

- Precision

$$PPV = \frac{TP}{TP + FP} \tag{15}$$

- Recall

$$TPR = \frac{TP}{TP + FN} \tag{16}$$

Oznaczenia użytych symboli:
TP - miara prawdziwie pozytywna (*true positive*)
TN - miara prawdziwie negatywna (*true negative*)
FP - miara fałszywie pozytywna (*false positive*)
FN - miara fałszywie negatywna (*false negative*)

3. Opis implementacji

Nasza aplikacja została utworzona w języku C# i jest to aplikacja konsolowa. Poniżej opisane zostały wszystkie klasy oraz dane zawarte w naszym projekcie:

— Klasa Program to klasa główna naszego programu. Jest swego rodzaju kontrolerem dla pozostałych klas. Znajduje się tutaj funkcja *main*, która rozpoczyna wykonywanie programu.

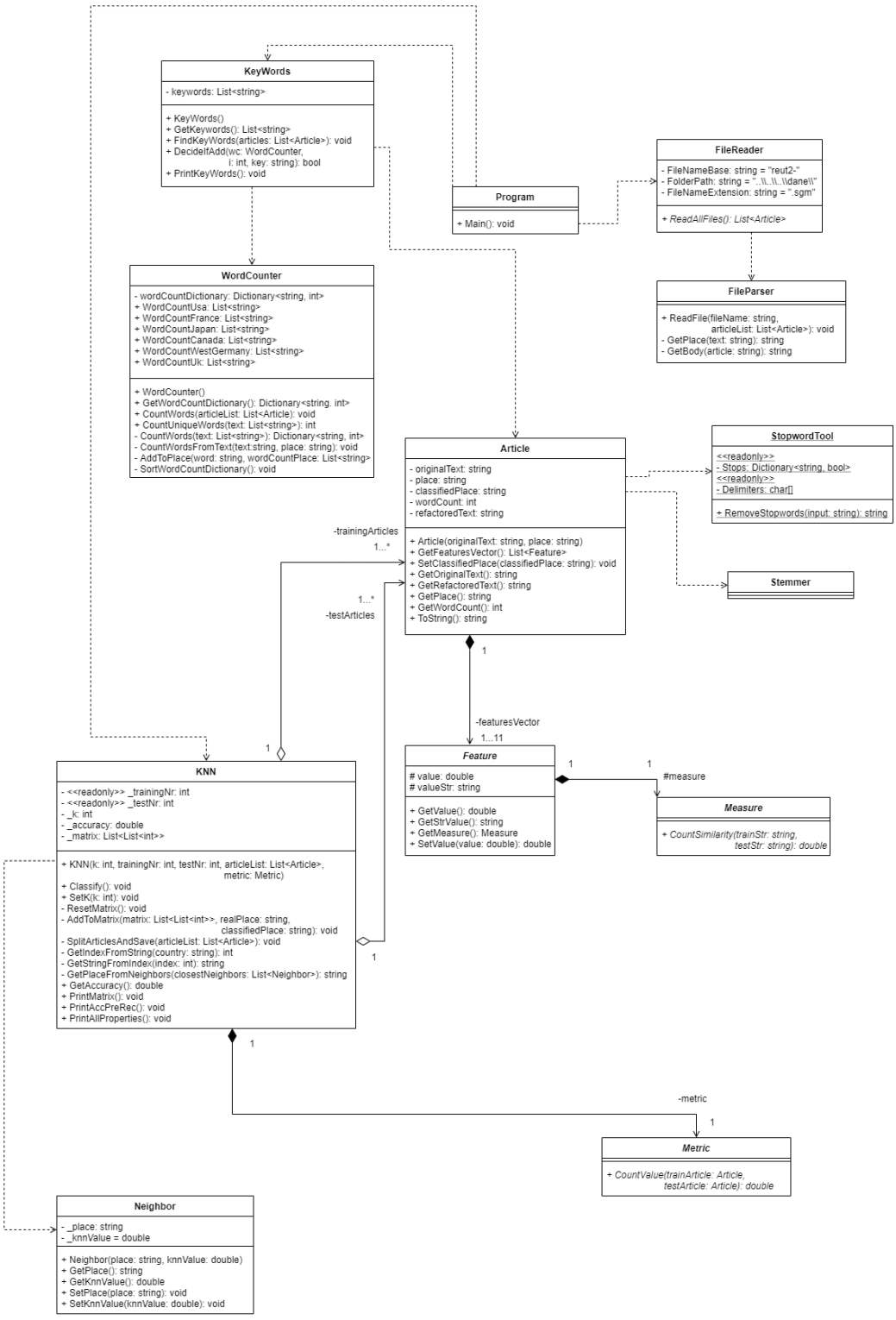
— W katalogu *dane* znajdują się wszystkie pliki z artyukłami, które są wykorzystywane do badań.
— Klasa Metric jest klasą abstrakcyjną. Odpowiada za obliczenia odległości tekstów. Po tej klasie dziedziczą klasy: EuclideanMetric, ChebyshewMetric oraz ManhattanMetric.
— Klasa Measure jest klasą abstrakcyjną. Po niej dziedziczą klasy *GeneralizedNGramsMeasure* i *NGramsMeasure*, które odpowiadają za obliczanie miar podobieństwa łańcuchów tekstowych.
— Klasa Feature jest klasą abstrakcyjną. Po niej dziedziczy 10 klas: Feature 1-10, które reprezentują każdą z 10 wyekstrahowanych przez nas cech.
— Klasa Stemmer to klasa, która odpowiada za stemizację tekstów. Została ona zapożyczona z [6]
— Klasa StopwordTool jest klasą odpowiedzialną za usuwanie słów znajdujących się na stopliście. Również została znaleziona i zapożyczona z Internetu ze strony [7]
— WordCounter jest używany do zliczania słów wszystkich artykułów i podania ich liczności. Potrzebny głównie do wyznaczenia słów kluczowych.
— Klasa KeyWords odpowiada za wyznaczenie 100 słów kluczowych. Metoda wyznaczania słów kluczowych została opisana w sekcji 2.
— Klasa FileReader odpowiada za otwieranie każdego pliku z artykułami
— FileParser to klasa odpowiedzialna za parsowanie danych z konkretnego pliku.
— Article to klasa reprezentująca artykuł. Zawiera takie cechy jak: tekst oryginalny, tekst przetworzony, *place*, *classifiedPlace*, wektor cech.
— Klasa Neighbor to klasa, która przechowuje artykuł oraz obliczoną wartość algorytmu k-NN dla konkretnego, obecnie sprawdzanego artykułu w algorytmie. Wykorzystujemy ją, aby znaleźć najbliższych k sąsiadów.
— KNN to klasa odpowiedzialna za algorytm k najbliższych sąsiadów.

Na rysunku 1 przedstawiony został wynik z konsoli po przykładowym uruchomieniu programu, natomiast na rysunku 2 przedstawiony został diagram UML naszego programu.

```
Settings: k=2 , training=30%, test=70%, metric=EuclideanMetric

 Place  Precision  Recall
usa  80,892  81,183
canada  8,921  11,524
france  0,621  0,595
japan  7,925  5,949
west-germany  9,459  5,578
uk  12,559  12,841
Accuracy: 66,415
```

Rysunek 1. Wynik z przykładowego uruchomienia programu.



Rysunek 2. Diagram UML.

4. Materiały i metody

Wykonana przez nas klasyfikacja została wykonana za pomocą wszystkich trzech metryk oraz dwóch miar podobieństwa. Każdy przypadek testowy był klasyfikowany dla dziesięciu różnych wartości k najbliższych sąsiadów: 2, 3, 4, 5, 7, 10, 13, 15, 20, 25.

Klasyfikacji dokonywaliśmy tylko na tych tekstach, które miały jedną z etykiet: *west-germany, usa, france, uk, canada, japan* i były to ich jedyne etykiety.

Dokonaliśmy pięciu różnych podziałów na dane testowe oraz treningowe:
— 30% dane treningowe, 70% dane testowe
— 50% dane treningowe, 50% dane testowe
— 70% dane treningowe, 30% dane testowe
— 80% dane treningowe, 20% dane testowe
— 85% dane treningowe, 15% dane testowe

Poniżej zostały opisane 4 wykonane przez nas eksperymenty.

4.1. Badanie zależności Accuracy od parametru k

W tym eksperymencie badaliśmy wpływ doboru parametru k na Accuracy. Program został uruchomiony dla 10 różnych wartości $k \in \{2, 3, 4, 5, 7, 10, 13, 15, 20, 25\}$.

Klasyfikacja tekstów została wykonana dla stałej wartości podziału zbioru cech na testowe i treningowe. Był to podział 50% dane treningowe, 50% dane testowe.

Metryką, jakiej użyliśmy była metryka euklidesowa.

4.2. Badanie wyników klasyfikacji w zależności od wartości proporcji podziału zbioru

W tym eksperymencie badaliśmy wpływ wartości proporcji podziału zbioru na Accuracy. Program został uruchomiony dla k=10.

Badane podziały były następujące:
— 30% dane treningowe, 70% dane testowe
— 50% dane treningowe, 50% dane testowe
— 70% dane treningowe, 30% dane testowe
— 80% dane treningowe, 20% dane testowe
— 85% dane treningowe, 15% dane testowe

Metryką, jakiej użyliśmy była metryka uliczna.

4.3. Badanie zależności Accuracy od wyboru metryki

W tym eksperymencie badaliśmy zależność Accuracy od wyboru metryki. Program został uruchomiony dla k=13. Podział na dane treningowe i testowe był stały i wynosił 70% treningowe i 30% testowe.

4.4. Badanie zależności Accuracy od wyboru podzbioru cech

W tym eksperymencie badaliśmy zależność Accuracy od wyboru podzbioru cech. Program został uruchomiony dla k=20. Metryka, jakiej użyliśmy to

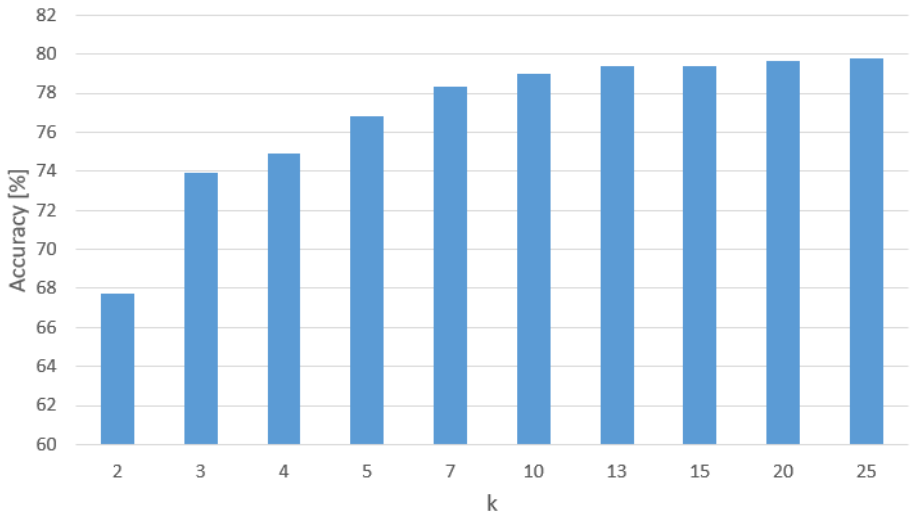
metryka Czebyszewa. Podział na dane treningowe i testowe był stały i wynosił 50% treningowe i 50% testowe. Podzbiory cech jakie badaliśmy były następujące:
— Wszystkie cechy
— $C_1, C_2, C_3, C_4, C_5, C_{11}$
— $C_6, C_7, C_8, C_9, C_{10}$
— $C_1, C_2, C_3, C_8, C_9, C_{10}$
— $C_4, C_5, C_6, C_7, C_{11}$

5. Wyniki

5.1. Badanie wyników klasyfikacji w zależności od parametru k

Parametr k	Accuracy [%]
2	67,705
3	73,921
4	74,900
5	76,799
7	78,312
10	79,024
13	79,365
15	79,410
20	79,632
25	79,795

Tabela 1. Zależność Accuracy od wartości k.

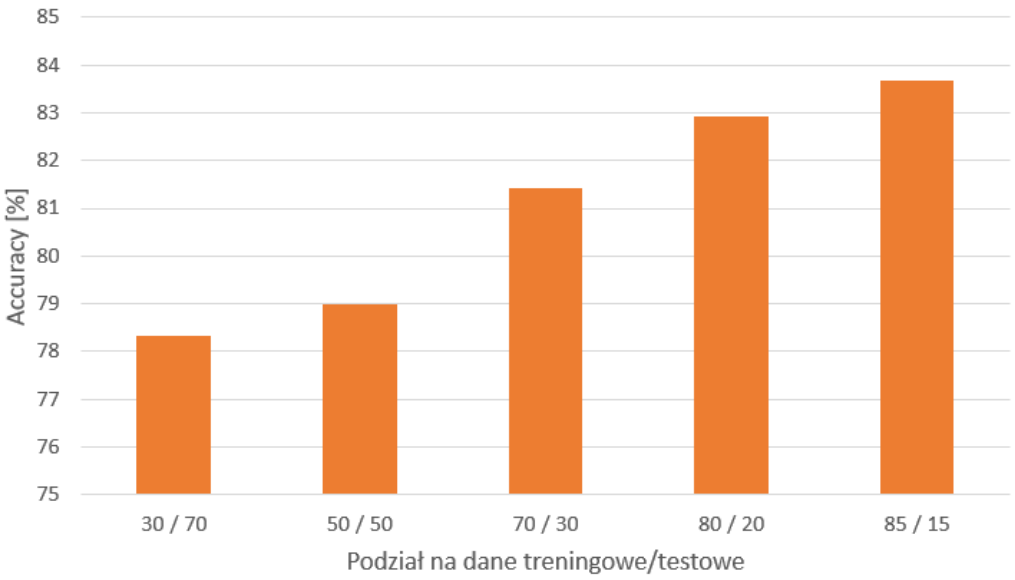


Rysunek 3. Wykres przedstawiający zależność Accuracy od wartości k (dane treningowe/testowe 50%/50%, Metryka euklidesowa).

5.2. Badanie wyników klasyfikacji w zależności od podziału na dane treningowe i testowe

Dane treningowe/testowe	Accuracy [%]
30/70	78,325
50/50	78,979
70/30	81,409
80/20	82,911
85/15	83,667

Tabela 2. Zależność Accuracy od pięciu wartości proporcji podziału zbioru dla k=10, metryka uliczna.

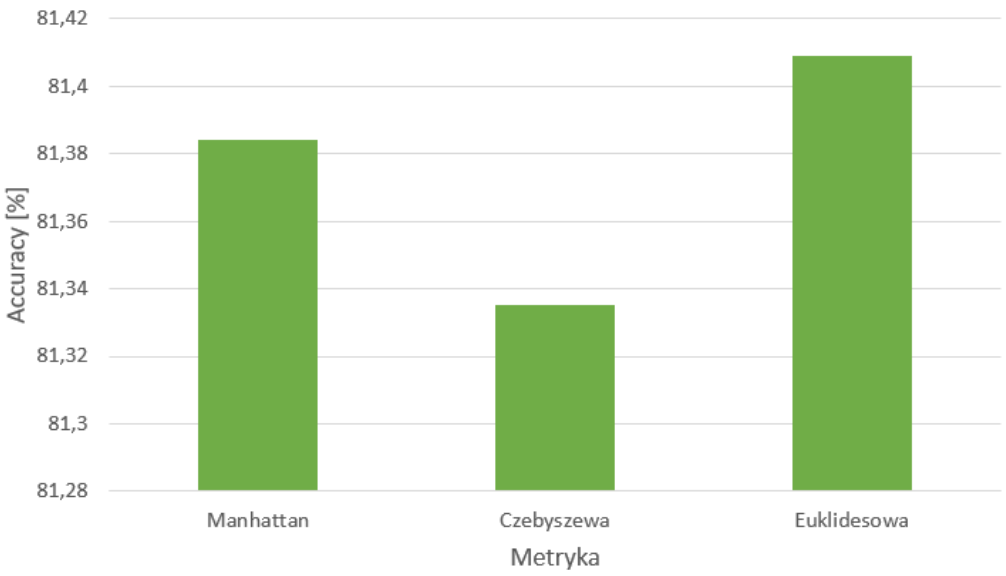


Rysunek 4. Wykres przedstawiający zależność Accuracy od pięciu wartości proporcji podziału zbioru, k=10, metryka uliczna.

5.3. Badanie zależności Accuracy od wyboru metryki

Metryka	Accuracy [%]
euklidesowa	81,409
Czebyszewa	81,335
uliczna	81,384

Tabela 3. Zależność Accuracy od wyboru metryki dla k=13 i podziału 70/30.

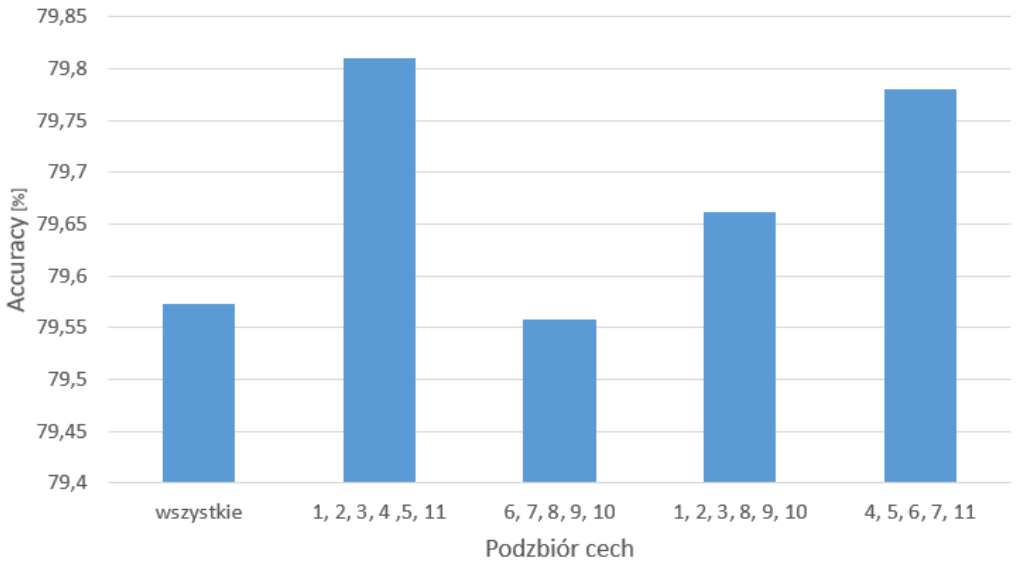


Rysunek 5. Wykres przedstawiający zależność Accuracy od wyboru metryki dla k=13 i podziału 70/30.

5.4. Badanie różnic w wyborze podzbioru cech

Podzbiór cech	Accuracy [%]
Wszystkie cechy	79,573
$C_1, C_2, C_3, C_4, C_5, C_{11}$	79,81
$C_6, C_7, C_8, C_9, C_{10}$	79,558
$C_1, C_2, C_3, C_8, C_9, C_{10}$	79,662
$C_4, C_5, C_6, C_7, C_{11}$	79,78

Tabela 4. Zależność Accuracy od wyboru podzbioru cech dla k=20, podziału 50/50 i metryki Czebyszewa.



Rysunek 6. Wykres przedstawiający zależność Accuracy od wyboru podzbioru cech, k=20, podział 50/50, metryka Czebyszewa.

6. Dyskusja

6.1. Wyniki klasyfikacji w zależności od parametru k

Dobór odpowiedniej wartości parametru k ma duży wpływ na wynik klasyfikacji. Im większa wartość k, tym większa skuteczność klasyfikacji. Można to zauważyć na rysunku 3, jak i również tabeli 1. Największy skok widać między $k = 2$ a $k = 3$. Natomiast przy wartościach wyższych niż 10 różnica jest bardzo niewielka. Najlepsze wyniki klasyfikacji były osiągane, gdy $k = 25$ - ok. 80%, natomiast najgorsze, gdy $k = 2$ - ok. 68%.

6.2. Wyniki klasyfikacji w zależności od podziału na dane treningowe i testowe

W przypadku podziału na dane testowe i treningowe, dla większej ilości danych treningowych algorytm był w stanie lepiej zaklasyfikować teksty i skuteczność klasyfikacji była wyższa. Dla badanych przez nas podziałów największą skuteczność osiągnął podział 85% dane treningowe i 15% dane testowe. Najgorsza jakość klasyfikacji została osiągnięta dla podziału 30% treningowe / 70% testowe.

Analizując rysunek 4 oraz tabelę 2 największy skok wystąpił między podziałami 50/50 a 70/30. W przypadku tego drugiego podziału algorytm był w stanie się lepiej nauczyć.

6.3. Zależność Accuracy od wyboru metryki

Wybór metryki w przypadku klasyfikowania artykułów wg kategorii *places* miał bardzo mały wpływ na wyniki klasyfikacji. Widać to na rysunku 5, a najlepiej w tabeli 3. Pomiędzy najlepszym wynikiem dla metryki euklidesowej, a najgorszym dla metryki Czebyszewa różnica była mniejsza niż 0.1 p.p. Możemy więc powiedzieć, że w przypadku algorytmu k-NN wybór metryki ma minimalny wpływ na wyniki klasyfikacji, jednak najlepszą metryką jest metryka euklidesowa.

6.4. Różnice w wyborze podzbioru cech

Jak można zauważyć na rysunku 6 oraz tabeli 4, najlepszą wartość klasyfikacji (79.81%) osiągnął podzbiór składający się z cech: $C_1, C_2, C_3, C_4, C_5, C_{11}$. Są to cechy zależne od słów kluczowych. Natomiast cechy niezależące od słów kluczowych, tj. $C_6, C_7, C_8, C_9, C_{10}$ osiągnęły wynik najgorszy (79.558%). Dla wszystkich cech ten wynik był podobny do wyniku dla cech 6-10. Różnice w skuteczności są niewielkie, lecz zauważalne.

7. Wnioski

— Liczba k sąsiadów ma spory wpływ na skuteczność klasyfikacji. Jednakże, zmiana metryki, bądź podziału na dane testowe i treningowe również ma wpływ na wynik klasyfikacji.
— Również istotny jest podział artykułów na dane testowe i treningowe.
— W przypadku zbyt małej ilości danych treningowych wystąpi zjawisko niedouczenia.
— W przypadku zbyt dużej ilości danych treningowych wystąpi zjawisko przeuczenia.
— Wybór metryki ma minimalny, bliski zeru wpływ na wyniki klasyfikacji.
— Najlepszymi cechami do klasyfikacji tekstów będą te cechy, które zależą od słów kluczowych, ale tylko wtedy, gdy te zostaną wybrane równomiernie dla każdej kategorii.

Literatura

[1] Niewiadomski, Adam. Methods for the Linguistic Summarization of Data: Applications of Fuzzy Sets and Their Extensions. Akademicka Oficyna Wydawnicza EXIT. Warszawa, 2008. ISBN 978-83-60434-40-6
[2] *https://ftims.edu.p.lodz.pl/pluginfile.php/132368/mod_folder/content/0/ksr-wyklad-2009.pdf?forcedownload=1* [dostęp 22.03.2020]
[3] *https://en.wikipedia.org/wiki/Taxicab_geometry* [dostęp 01.04.2020]
[4] *https://en.wikipedia.org/wiki/Chebyshev_distance* [dostęp 01.04.2020]
[5] *https://en.wikipedia.org/wiki/Euclidean_distance* [dostęp 01.04.2020]
[6] *https://tartarus.org/martin/PorterStemmer/csharp.txt* [dostęp 22.03.2020]
[7] *https://www.dotnetperls.com/stopword-dictionary* [dostęp 22.03.2020]
[8] *http://home.agh.edu.pl/ horzyk/lectures/miw/KNN.pdf* [dostęp 22.03.2020]
[9] *https://pl.wikipedia.org/wiki/Tablica_pomy%C5%82ek* [dostęp 01.04.2020]