Rozdział 1

Odległości na przestrzeni ciągów znaków

1.1. Podstawowe definicje

Definicja 1.1. *Niech $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ będzie skończonym uporządkowanym zbiorem o liczności $|\Sigma|$, zwanym* alfabetem. Napisem *nazywamy skończony ciąg znaków z Σ. Zbiór wszystkich napisów o długości n nad Σ jest oznaczony przez Σ^n, podczas gdy przez $\Sigma^* = \bigcup_{n=1}^{\infty} \Sigma^n$ rozumiemy zbiór wszystkich napisów utworzonych ze znaków z Σ [4].*

O ile nie podano inaczej, używamy zmiennych s, t, u, v, w, x, y jako oznaczenie napisów oraz a, b, c do oznaczenia napisów jednoznakowych albo po prostu *znaków*. Pusty napis jest oznaczany przez ε. Przez $|s|$, dla każdego napisu $s \in \Sigma^*$, rozumiemy jego długość, czyli liczbę znaków w napisie. Ciąg napisów i/lub znaków oznacza ich złączenie, np. *stu* to napis powstały ze złączenia napisów s, t oraz u, natomiast *abc*, to napis powstały ze złączenia znaków a, b oraz c [4]. Dla rozróżnienia napisów od zmiennych reprezentujących napis, te pierwsze oznaczamy pismem maszynowym, np. `napis`.

Poprzez s_i rozumiemy i-ty znak z napisu s, dla każdego $i \in \{1, \ldots, |s|\}$. Podciąg kolejnych przylegających do siebie znaków z napisu nazywamy *podnapisem*. Podnapisem napisu s, który zaczyna się od i-tego znaku, a kończy na j-tym znaku, oznaczamy przez $s_{i:j}$, tj. $s_{i:j} = s_i s_{i+1} \ldots s_j$ dla $i \leq j$. Zakładamy również, że jeśli $j < i$, to $s_{i:j} = \varepsilon$ [4, 15].

Definicja 1.2. *Załóżmy, że napis s jest reprezentacją złączenia trzech, być może pustych, podnapisów w, x i y, tj. $s = wxy$. Wówczas podnapis w nazywamy* przedrostkiem, *natomiast podnapis y – przyrostek [4].*

Definicja 1.3. *Podnapis złożony z kolejnych, przylegających do siebie, znaków napisu, o ustalonej długości q jest nazywany q-*gramem. *q-gramy o q równym jeden, dwa lub trzy mają specjalne nazwy:* unigram, bigram *i* trigram. *Jeśli $q > |s|$, to q-gramy napisu s są napisami pustymi [4].*

Przykład 1.1. Niech Σ będzie alfabetem złożonym z 26 małych liter alfabetu łacińskiego oraz niech $s =$ `ela`. Wówczas mamy $|s| = 3$, $s \in \Sigma^3$ oraz $s \in \Sigma^*$. Co więcej, mamy $s_1 =$ `e`,

$s_2 =$ `l`, $s_3 =$ `a`. Podnapis $1:2$ napisu s to $s_{1:2} =$ `el`. W napisie tym mamy do czynienia jedynie z q-gramami o q równym jeden, dwa oraz trzy, odpowiednio: `e`, `l`, `a`; `el`, `la` oraz `ela`.

We wszystkich przykładach niniejszego rozdziału zakładamy, jeśli nie podano inaczej, że alfabet składa się z 32 liter polskiego alfabetu oraz liter `q`, `v` i `x`.

1.2. Odległości na przestrzeni ciągów znaków

W niniejszym podrozdziale zajmiemy się odległościami na przestrzeni ciągów znaków, tj. funkcjami $d : \Sigma^* \times \Sigma^* \to [0, \infty)$. W literaturze można znaleźć wiele różnych funkcji tego typu, które różnią się genezą powstania, podejściem do problemu oraz zastosowaniami. W pracy zajmiemy się jednak odległościami, która można podzielić na trzy garupy:

- oparte na operacjach edycyjnych (*edit operations*),
- oparte na q-gramach,
- miary heurystyczne.

BLA BLA JAKIEŚ LANIE WODY O METRYKACH Pierwszy rodzaj odległości jest najczęściej używany w algorytmach zajmujących się optymalnym dopasowaniem, dlatego też poświęcimy mu największą część niniejszego rozdziału. Odległości oparte na q-gramach ... (TUTAJ COŚ O NICH). Natomiast miary heurystyczne są rzadko stosowane, będąc zazwyczaj używane w konkretnych przypadkach. Miary te miały jednak swój wkład w historię optymalizacji [NIE!] napisów, zatem pokrótce opiszemy je pod koniec tego rozdziału.

1.2.1. Odległości oparte na operacjach edycyjnych

HISTORIA ODLEGLOSCI EDYCYJNYCH?

Ścieżka edycyjna i bazowe operacje edycyjne. *Odległość edycyjna* $ED(s, t)$ między dwoma napisami s i t to minimalna liczba operacji edycyjnych potrzebna do przetworzenia s w t (i ∞, gdy taki ciąg nie istnieje) [8]. *Ścisłą odległością edycyjną* nazywamy minimalną liczbę nienakładających się operacji edycyjnych, które pozwalają przekształcić jeden napis w drugi, i które nie przekształcają dwa razy tego samego podnapisu [4].

Napis może zostać przetworzony w drugi poprzez wykonanie na nim ciągu przekształceń jego podnapisów. Ten ciąg nazywany jest *ścieżką edycyjną (śladem edycji?)*, podczas gdy przekształcenia są nazywane *bazowymi operacjami edycyjnymi*. Bazowe operacje edycyjne, które polegają na przekształceniu napisu s w napis t, są oznaczane przez $s \to t$. Zbiór wszystkich bazowych operacji edycyjnych oznaczamy przez $\mathbb{B}$ [4].

Bazowe operacje edycyjne są zazwyczaj ograniczone do:

- usunięcie znaku: `l` $\to \varepsilon$, tj. usunięcie litery `l` , np. `ela` $\to$ `ea`,
- wstawienie znaku: $\varepsilon \to$ `k`, tj. wstawienie litery `k`, np. `ela` $\to$ `elka`,
- zamiana znaku: `e` $\to$ `a`, tj. zamiana litery `e` na `a`, np. `ala` $\to$ `ela`,
- transpozycja: `el` $\to$ `le`, tj. przestawienie dwóch przylegających liter `e` i `l`, np. `ela` $\to$ `lea`.

Często transpozycja znaków nie należy do zbioru operacji bazowych, jako że można ją zastąpić usunięciem i wstawieniem znaku. W niniejszej pracy jednak, operacja ta należy do zbioru operacji bazowych.

Własność 1.1. *Zakładamy, że* $\mathbb{B}$ *spełnia następujące własności [4]:*

- *jeśli* $s \to t \in \mathbb{B}$*, to odwrotna operacja* $t \to s$ *również należy do* $\mathbb{B}$*;*
- $a \to a \in \mathbb{B}$ *(operacja identycznościowa dla jednego znaku należy do* $\mathbb{B}$*);*
- *zbiór* $\mathbb{B}$ *jest zupełny: dla dwóch dowolnych napisów* s *i* t*, istnieje ślad edycji, który przekształca* s *w* t*.*

Zauważmy, że zbiór $\mathbb{B}$ nie musi być skończony.

Odległość edycyjna. Podobieństwo dwóch napisów może być wyrażone jako długość ścieżki edycyjnej, dzięki której jeden napis zostaje przekształcony w drugi:

Definicja 1.4. *Mając dany zbiór bazowych operacji edycyjnych,* odległość edycyjna $ED(s,t)$ *jest równa długości najkrótszej ścieżki edycyjnej, która przekształca napis* s *w napis* t*. Najkrótsza ścieżka, która przekształca napis* s *w napis* t *jest nazywana* optymalną *ścieżką edycyjną [4].*

Przykład 1.2. JAKIŚ PRZYKŁĄD SCIEZKI I OPTYMALNEJ SCIEZKI.

Przykładowe odległości edycyjne: Hamminga, najdłuższego wspólnego podnapisu (*longest common substring*), Levenshteina, optymalnego dopasowania napisów (*optimal string alignment*), Damareu-Levenshteina. Odległości te różnią się zbiorem bazowych operacji edycyjnych. Jeśli w zbiorze tym znajduje się tylko zamiana znaków, to mamy do czynienia z odległością Hamminga. Gdy zbiór bazowych operacji edycyjnych zawiera wstawienia i usunięcia znaków, to jest to odległość najdłuższego wspólnego podnapisu. Gdyby $\mathbb{B}$ powiększyć o zamianę znaków, to otrzymamy odległość Levenshteina. Dwie ostatnie odległości, tj. optymalnego dopasowania napisów oraz Damareu-Levenshteina, mają w zbiorze bazowych operacji edycyjnych usunięcie, wstawienie, zamianę oraz transpozycję znaków. Formalne definicje powyższych funkcji znajdują się w dalszej części niniejszego rozdziału.

Definicja odległości edycyjnej może być również interpretowana jako minimalny koszt, dzięki któremu przekształcamy jeden napis w drugi. Definicję można uogólnić na dwa sposoby. Po pierwsze, bazowe operacje edycyjne mogą mieć przydzielone koszty (wagi) $\delta(a \to b)$ [16]. Zazwyczaj koszt każdej operacji wynosi jeden, jednak można, na przykład, nadać transpozycji mniejszy koszt niż operacji wstawienia znaku. Dalej, można rozszerzyć funkcję kosztu δ na ścieżkę edycyjną $E = a_1 \to b_1, a_2 \to b_2, \ldots, a_{|E|} \to b_{|E|}$ przez $\delta(E) = \sum_{i=1}^{|E|} \delta(a_i \to b_i)$ [4]. Odtąd przez odległość między napisem s a napisem t będziemy rozumieć minimalny ze wszystkich możliwych kosztów ścieżek przekształcających s w t. Odległości zdefiniowane w ten sposób zazwyczaj są nazywane *uogólnionymi* odległościami edycyjnymi.
Po drugie, zbiór operacji edycyjnych $\mathbb{B}$ może zostać rozszerzony o ważone zamiany (substytucje) (pod)napisów, zamiast operacji edycyjnych wykonywanych na pojedynczych znakach [13]. Odległości zdefiniowane w ten sposób zazwyczaj są nazwyane *rozszerzonymi* odległościami

edycyjnymi. Przykładowo, $\mathbb{B}$ może zawierać operację $\texttt{x} \to \texttt{ks}$ o koszcie jednostkowym. Wówczas rozszerzona odległość pomiędzy napisami `xero` i `ksero` wynosi jeden, podczas gdy standardowa (zwykła, nierozszerzona) odległość wyniosłaby dwa [4].

Definicja 1.5. *Mając dany zbiór bazowych operacji edycyjnych $\mathbb{B}$ oraz funkcję δ, która nadaje koszt wszystkim bazowym operacjom edycyjnym z $\mathbb{B}$, uogólniona odległość edycyjna między napisami s i t jest zdefiniowana jako minimalny spośród kosztów wszystkich możliwych ścieżek edycyjnych, które przekształcają s w t [4].*

[KONFLIKT Z OSTATNIM ZDANIEM POPRZEDNIEGO AKAPITU - UOGOLNIONYMI ODL. ED.] Zazwyczaj koszt pojedynczej operacji z $\mathbb{B}$ jest równy jeden. Czasem jednak nadaje się poszczególnym operacjom różne koszty, dając np. transpozycji mniejszą wagę niż wstawieniu znaku. Gdy koszt wszystkich operacji jest równy jeden, to mówimy po prostu o odległości edycyjnej, natomiast gdy różne operacje mają różne wagi, to mówimy o *uogólnionej* odległości edycyjnej.

Własność 1.2. *Zakładamy, że funkcja kosztu $\delta(s \to t)$ ma następujące własności [4]:*

- $\delta(s \to t) \in \mathbb{R}$ *(koszt operacji jest liczbą rzeczywistą),*
- $\delta(s \to t) = \delta(t \to s)$ *(symetria),*
- $\delta(s \to t) \geq 0,\ \delta(s \to s) = 0$ *i* $\delta(s \to t) = 0 \Rightarrow s = t$ *(dodatnia określoność ??),*
- $\forall \gamma > 0$ *zbiór bazowych operacji* $\{s \to t \in \mathbb{B} | \delta(s \to t) < \gamma\}$ *jest skończony (skończoność podzbioru bazowych operacji, których koszt jest ograniczony z góry).*

Zauważmy, że ostatnia własność jest zawsze spełniona dla skończonego zbioru $\mathbb{B}$.

Twierdzenie 1.3. *Z własności 1.1 i 1.2 wynika, że:*

- *dla każdych dwóch napisów s i t, istnieje ścieżka o minimalnym koszcie, tj. dobrze zdefiniowana odległość edycyjna z s do t [4],*
- *ogólna odległość edycyjna z definicji 1.5 jest metryką [16].*

Dowód. Żeby udowodnić, że $ED(s,t)$ jest metryką, musimy pokazać, że $ED(s,t)$ istnieje, jest dodatnio określona, symetryczna oraz subaddytywna (tj. spełnia nierówność trójkąta).

Z własności 1.2 wynika, że funkcja kosztu jest nieujemna (JA TEGO NIW WIDZE) i że tylko identyczność ma koszt równy zero. Stąd, bez utraty ogólności, możemy rozważyć jedynie takie ścieżki edycyjne, które nie zawierają operacji identycznościowych. Zatem, jeśli $s = t$, to jedyna optymalna ścieżka (która nie zawiera operacji identycznościowych) jest pusta i ma zerowy koszt. Jeśli $s \neq t$, to z zupełności zbioru bazowych operacji edycyjnych wynika, że istnieje jedna lub więcej ścieżek edycyjnych, które przekształcają s w t. Wszystkie te ścieżki składają się z operacji edycyjnych o ściśle dodatnim koszcie.

Niech γ będzie kosztem ścieżki przekształcającej s w t. Rozważmy zbiór A ścieżek edycyjnych, które przekształcają s w t i których koszt jest ograniczony z góry przez γ. Zbiór A jest niepusty i składa się z operacji edycyjnych o dodatnim koszcie mniejszym niż γ. Zbiór operacji bazowych, których koszt jest ograniczony z góry przez γ jest skończony, co dowodzi, że zbiór A jest również skończony. Ponieważ A jest niepusty i skończony, to ścieżki edycyjne

o mininalnym (dodatnim) koszcie istnieją i należą do A. Stąd, $ED(s,t) > 0$ dla $s \neq t$, tj. odległość edycyjna jest dodatnio określona.

Aby udowodnić symetrię odległości edycyjnej, rozważmy optymalną ścieżkę E, która przekształca s w t, oraz odpowiadającą jej odwrotną ścieżkę E_r, która przekształca t w s. Równość ich kosztów $\delta(E) = \delta(E_r)$ wynika z symetrii funkcji kosztu i symetrii zbioru operacji bazowych $\mathbb{B}$.

Aby pokazać subaddytywność, rozważmy optymalną ścieżkę E_1, która przekształca s w t, optymalną ścieżkę E_2, która przekształca t w u, oraz złożenie ścieżek E_1E_2, które przekształca s w u. Z tego, że $\delta(E_1E_2) = \delta(E_1) + \delta(E_2) = ED(s,t) + ED(t,u)$ oraz $\delta(E_1E_2) \geq ED(s,u)$ (gdyż E_1E_2 nie musi być optymalną ścieżka, przekształcającą s w u) wynika, że $ED(s,t) + ED(t,u) \geq ED(s,u)$. ■

Odległość edycyjna jest metryką, nawet gdy funkcja kosztu δ nie jest subaddytywna. Co więcej, ponieważ ciąg nakładających się operacji, które przekształcają s w t, mogą mieć mniejszy koszt niż $\delta(s \to t)$, $\delta(s \to t)$ może być większe niż $ED(s,t)$. Rozważmy, na przykład, następujący alfabet: {`a, b, c`}, gdzie symetria i brak subaddytywności funkcji δ jest zdefiniowana następująco:

$$\begin{aligned}
&\delta(\texttt{a} \to \texttt{c}) = \delta(\texttt{b} \to \texttt{c}) = 1\\
&\delta(\texttt{a} \to \varepsilon) = \delta(\texttt{b} \to \varepsilon) = \delta(\texttt{c} \to \varepsilon) = 2\\
&\delta(\texttt{a} \to \texttt{b}) = 3
\end{aligned}$$

Można zobaczyć, że $3 = \delta(\texttt{a} \to \texttt{b}) > \delta(\texttt{a} \to \texttt{c}) + \delta(\texttt{c} \to \texttt{b}) = 2$. Stąd optymalna ścieżka edycyjna $(\texttt{a} \to \texttt{c}, \texttt{c} \to \texttt{b})$ przekształca `a` w `b` z kosztem równym 2.

Ścisła odległość edycyjna. Subaddytywność odległości edycyjnej pozwala używać metod właściwych przestrzeniom metrycznym. Niemniej jednak, problem minimalizacji zbioru nakładających się operacji edycyjnych, może być trudny. Aby zrównoważyć złożoność obliczeniową, zazwyczaj używana jest funkcja podobieństwa, zdefiniowana jako minimum kosztu *ścisłej ścieżki edycyjnej*. Ta ostatnia nie zawiera nakładających się na siebie operacji edycyjnych i nie modyfikuje tego samego podnapisu więcej niż raz. Odpowiająca jej odległość edycyjna nazywana jest *ścisłą odległością edycyjną* [4].

Lemat 1.4. *Dowolna nieścisła odległość edycyjna ogranicza z dołu odpowiadającą jej ścisłą odległość edycyjną [4].*

Lemat 1.5. *Ścisła odległość Levenshteina o jednostkowym koszcie operacji bazowych jest równa nieścisłej odległości Levenshteina o jednostkowym koszcie operacji bazowych [4].*

Powyższe wynika natychmiast z obserwacji, że optymalna ścieżka edycyjna zawiera jednoznakowe usunięcia, wstawienia oraz zamiany, które nigdy nie modyfikują podnapisu więcej niż raz.

Lemat 1.6. *Nieścisła odległość Damerau-Levenshteina oraz ścisła odległość Damerau-Levenshteina są różnymi funkcjami. Co więcej, ścisła odległość Damerau-Levenshteina nie jest metryką, gdyż nie jest subaddytywna [4].*

Dowód. Ścisła odległość Damerau-Levenshteina traktuje transpozycję (tj. zamianę dwóch przylegających do siebie znaków) jako bazową operację edycyjną. Aby udowodnić lemat podamy przykład, w którym zakaz modyfikacji znaków już stransponowanych odróżnia odległość Damerau-Levenshteina od ścisłej odległości Damerau-Levenshteina [4]. ■

Rozważmy napisy `ab`, `ba` oraz `acb`. Z jednej strony, najkrótsza nieścisła ścieżka edycyjna, która przekształca `ba` w `acb`, tj. ($\texttt{ba} \to \texttt{ab}, \varepsilon \to \texttt{c}$) zawiera dwie operacje: najpierws zamienia znaki `a` i `b`, a następnie wstawia `c` pomiędzy nie. Zauważmy, że wstawienie przekształca już transformowany napis. Jednakowoż, jeśli kolejne przekształcenia tego samego podnapisu są wykluczone, to najkrótsza ścieżka edycyjna, która przekształca `ba` w `acb`, składa się z trzech operacji edycyjnych, np. ($\texttt{ba} \to \varepsilon, \varepsilon \to \texttt{c}, \varepsilon \to \texttt{b}$). Stąd, nieścisła odległość edycyjna jest równa dwa, podczas gdy ścisła odległość wynosi trzy [4].

Ścisła odległość Damerau-Levenshteina nie spełnia nierówności trójkąta, gdyż

$$\texttt{ba} \xrightarrow[1]{\textit{transp. b i a}} \texttt{ab} + \texttt{ab} \xrightarrow[1]{\textit{wst.c}} \texttt{acb},$$

natomiast

$$\texttt{ba} \xrightarrow[1]{\textit{us. b}} \texttt{a} \xrightarrow[1]{\textit{wst. c}} \texttt{ac} \xrightarrow[1]{\textit{wst. b}} \texttt{acb},$$

zatem

$$2 = ED(\texttt{ba}, \texttt{ab}) + ED(\texttt{ab}, \texttt{acb}) \leq ED(\texttt{ba}, \texttt{acb}) = 3.$$

Ponieważ ścisła i nieścisła odległość Damerau-Levenshteina są różnymi funkcjami, tę pierwszą nazywa się często *odległością optymalnego dopasowania napisów*. Od tego momentu w niniejszej pracy ścisłą odległość Damerau-Levenshteina nazywamy odległością optymalnego dopasowania napisów, natomiast nieścisłą odległość Damerau-Levenshteina nazywamy odległością Damerau-Levenshteina [15].

Optymalne dopasowanie.

Niech napisy s i t zostaną podzielone na tę samą liczbę, być może pustych, podnapisów: $s = s_1 s_2 \dots s_l$ i $t = t_1 t_2 \dots t_l$, takich, że $s_i \to t_i \in \mathbb{B}$. Co więcej, zakładamy, że s_i i t_j nie mogą być puste dla $i = j$. Mówimy, że ten podział definiuje *dopasowanie* $A = (s_1 s_2 \dots s_l, t_1 t_2 \dots t_l)$ pomiędzy napisami s i t, w którym podnapis s_i jest dopasowany do podnapisu t_i [4].

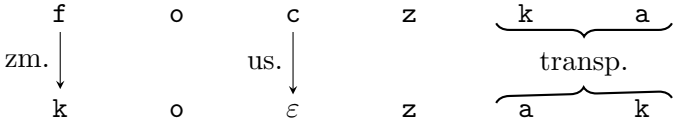
Dopasowanie reprezentuje ścisłą ścieżkę edycyjną $E = s_1 \to t_1, s_2 \to t_2, \dots, s_l \to t_l$. Definiujemy *koszt dopasowania* A jako koszt odpowiadającej mu ścieżki edycyjnej i oznaczamy go przez $\delta(A)$:

$$\delta(A) = \sum_{i=1}^{l} \delta(s_i \to t_i) \tag{1.1}$$

Optymalne dopasowanie to dopasowanie o najmniejszym koszcie [4].

Przykład 1.3. Przykład optymalnego dopasowania pomiędzy słowami `foczka` i `kozak` prezentuje rys. 1.1. Odpowiadająca mu ścieżka edycyjna składa się z zamiany $\texttt{f} \to \texttt{k}$, usunięcia $\texttt{c} \to \varepsilon$ oraz transpozycji $\texttt{ka} \to \texttt{ak}$.

Warto zauważyć, że istnieje różnowartościowe (1-1) mapowanie między zbiorem ścisłych ścieżek edycyjnych i zbiorem [optymalnych?] dopasowań: każda ścisła ścieżka edycyjna o minimalnym koszcie reprezentuje dopasowanie o najmniejszym koszcie i odwrotnie. Stąd można



Rysunek 1.1: Przykład optymalnego dopasowania między napisami `foczka` i `kozak`.

zastąpić problem znalezienia optymalnej ścisłej odległości edycyjnej poprzez problem znalezienia optymalnego dopasowania, co też zastosujemy dalej [4].

Obliczanie odległości edycyjnej. Główną zasadą dynamicznego algorytmu, liczącego koszt optymalnego dopasowania, jest wyrażenie kosztu dopasowania pomiędzy napisami s i t, używając kosztu dopasowania ich przedrostków. Rozważmy prefiks $s_{1:i}$ o długości i i przedrostek $t_{1:j}$ o długości j, odpowiednio napisów s i t. Załóżmy, że $A = (s_1 s_2 \dots s_l, t_1 t_2 \dots t_l)$ jest optymalnym dopasowaniem między $s_{1:i}$ i $t_{1:j}$, którego koszt oznaczamy przez $C_{i,j}$ [4].

Używając równania 1.1 oraz definicji optymalnego dopasowania, łatwo pokazać, że $C_{i,j}$ może zostać policzone przy użyciu następującej ogólnej rekurencji [13]:

$$\begin{aligned} C_{0,0} &= 0 \\ C_{i,j} &= \min\{\delta(s_{i':i} \to t_{j':j}) + C_{i'-1,j'-1} | s_{i':i} \to t_{j':j} \in \mathbb{B}\}. \end{aligned} \tag{1.2}$$

Można zauważyć, że:

- koszt dopasowania napisów s i t jest równy $C_{|s|,|t|}$;
- wszystkie optymalne dopasowania mogą zostać wyznaczone przez odwracanie rekurencji 1.2 (przechodzenie od tyłu), tj. obliczanie najpierw $C_{0,0}$, następnie $C_{1,1}$ itd.

Rozważmy teraz odległość Hamminga, gdzie $s_{i':i} \to t_{j':j}$ to zamiany znaków o koszcie równym jeden. Stąd,

$$\delta(s_{i':i} \to t_{j':j}) = [s_{i':i} \neq t_{j':j}] \tag{1.3}$$

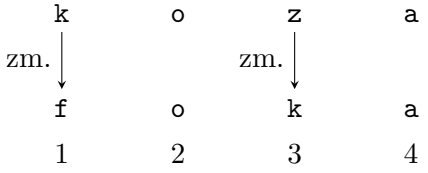
gdzie $[X]$ jest równe jeden, gdy warunek X jest spełniony, zero w przeciwnym przypadku. Co więcej, w tym przypadku możliwa jest tylko jedna kombinacja i' oraz j', mianowicie $i' = i$ oraz $j' = j$. Dalej, odległość ta jest zdefiniowana jedynie dla $|s| = |t|$, zatem $C_{i,j}$ może być policzone jedynie dla $i = j$. Wówczas definicja odległości Hamminga nie jest rekurencyjna i można ją zapisać następująco:

Definicja 1.6. Odległością Hamminga *nazywamy [5]:*

$$d_{\text{hamming}}(s,t) = \begin{cases} \sum_{i=1}^{|s|} \delta(s_i \to t_i) = \sum_{i=1}^{|s|} [s_i \neq t_i], & \textit{gdy } |s| = |t|, \\ \infty, & \textit{w przeciwnym przypadku,} \end{cases}$$

Odległość Hamminga zlicza liczbę indeksów (p. rys. 1.2), na których dwa napisy mają różny znak. Odległość ta przyjmuje wartości ze zbioru $\{0, \dots, |s|\}$, gdy $|s| = |t|$, natomiast jest równa nieskończoności, gdy napisy mają różne długości.

[PIĘKNY RYSUNEK]



Rysunek 1.2: Przykład dopasowania przy pomocy odległości Hamming między napisami `koza` i `foka`.

Przykład 1.4. Odległość Hamminga między słowami `koza` i `foka` wynosi $d_{\text{hamming}}(\texttt{koza}, \texttt{foka}) = 2$, natomiast między słowami `kozak` i `foczka` wynosi ona $d_{\text{hamming}}(\texttt{kozak}, \texttt{foczka}) = \infty$, gdyż $|\texttt{kozak}| \neq |\texttt{foczka}|$.

Rozważmy teraz odległość najdłuższego wspólnego podnapisu, gdzie $s_{i':i} \to t_{j':j}$ to wstawienia i usunięcia znaków o koszcie równym jeden. Wówczas istnieją dwie kombinacje i' oraz j' z ogólnej rekurencji 1.3, odpowiadające usunięciu i wstawieniu, odpowiednio:

- $i' = i - 1$ oraz $j' = j$,
- $i' = i$ oraz $j' = j - 1$.

Uwzględniając powyższe uproszczenia, możemy następująco przepisać ogólną postać rekurencji 1.2 dla odległości najdłuższego wspólnego podnapisu [NIE WIEM CZY TO JEST DOBRZE!!!!]:

$$C_{i,j} = \min \begin{cases} 0, & \text{gdy } i = j = 0 \\ C_{i-1,j} + 1, & \text{gdy } i > 0 \\ C_{i,j-1} + 1, & \text{gdy } j > 0 \end{cases}$$

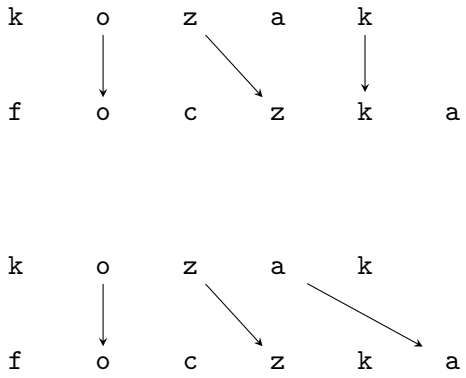
Odległość najdłuższego wspólnego podnapisu przyjmuje wartości ze zbioru $\{0, |s| + |t|\}$, przy czym maksimum jest osiągane, gdy s i t nie mają ani jednego wspólnego znaku. Odległość tę oznaczamy przez d_{lcs}.

Przykład 1.5. Odległość najdłuższego wspólnego podnapisu między napisami `kozak` i `foczka` wynosi: $d_{lsc}(\texttt{kozak}, \texttt{foczka}) = 5$, bo $\texttt{kozak} \xrightarrow[1]{us.\ k} \texttt{ozak} \xrightarrow[1]{us.\ a} \texttt{ozk} \xrightarrow[1]{wst.\ f} \texttt{fozk} \xrightarrow[1]{wst.\ c} \texttt{foczk} \xrightarrow[1]{wst.\ a} \texttt{foczka}$.

Powyższy przykład pokazuje, że w ogólności nie ma unikalnej najkrótszej drogi transformacji jednego napisu w drugi, gdyż można zamienić kolejność usuwania (lub wstawiania) znaków i również uzyskać odległość równą 5. Można również usunąć z napisu znak `k` zamiast `a`, otrzymując taką samą odległość między napisami.

Jak sugeruje nazwa, odległość najdłuższego wspólnego podnapisu, ma też inną interpretację. Poprzez wyrażenie *najdłuższy wspólny podnapis* rozumiemy najdłuższy ciąg utworzony przez sparowanie znaków z s i t nie zmieniając ich porządku. Wówczas odległość ta jest rozumiana jako liczba niesparowanych znaków z obu napisów. W powyższym przykładzie może to być zwizualizowane następująco (rys. 1.3):

[PIĘKNY RYSUNEK]



Rysunek 1.3: Przykład odległości najdłuższego wspólnego podnapisu między napisami `kozak` i `foczka`.

Jak widać na rysunku, znaki `k`, `a`, `f`, `c` i `a` w pierwszym przypadku oraz `k`, `k`, `f`, `c` i `k` w drugim, pozostają bez pary, dając odległość równą 5.

Przejdźmy do odległości Levenshteina, d_{lv}. Odległość ta dopuszcza, oprócz usunięć i wstawień, także zamiany znaków. Istnieją zatem trzy kombinacje i' oraz j' z ogólnej rekurencji 1.3:

- $i' = i - 1$ oraz $j' = j$,
- $i' = i$ oraz $j' = j - 1$,
- $i' = i - 1$ oraz $j' = j - 1$.

Stąd ogólna postać rekurencji 1.2 dla odległości Levenshteina może zostać przepisana następująco:

$$C_{i,j} = \min \begin{cases} 0, & \text{gdy } i = j = 0 \\ C_{i-1,j} + 1, & \text{gdy } i > 0 \\ C_{i,j-1} + 1, & \text{gdy } j > 0 \\ C_{i-1,j-1} + [s_i \neq t_j], & \text{gdy } i, j > 0 \end{cases} \tag{1.4}$$

Odległość Levenshteina oznaczamy przez d_{lv}.

Przykład 1.6. Odległość Levenshteina między napisami `kozak` i `foczka` wynosi: $d_{\mathrm{lv}}(\texttt{kozak}, \texttt{foczka}) = 4$, bo $\texttt{kozak} \xrightarrow[1]{zm.\ k\ na\ f} \texttt{fozak} \xrightarrow[1]{wst.\ c} \texttt{foczak} \xrightarrow[1]{zm.\ a\ na\ k} \texttt{foczkk} \xrightarrow[1]{zm.\ k\ na\ a} \texttt{foczka}$.

Powyższy przykład ilustruje dodatkową elastyczność w porównaniu do odległości najdłuższego wspólnego podnapisu, bowiem daje ona mniejszą wartość odległości między napisami, jako że w przypadku pierwszego znaku potrzebujemy jedynie zamiany, zamiast wstawienia i usunięcia [15]. Co więcej, ścieżka edycyjna między tymi słowami może być inna i zawierać usunięcie, i wstawienie zamiast dwóch ostatnich zamian znaków.

[ZMIENIC NA KOZAKA I FOCZKE!!] Przypomnijmy, że mówimy o uogólnionej odległości, gdy zmienimy koszty poszczególnych operacji na różne od jeden. Gdy za koszt przyjmiemy np. $(0.1, 1, 0.3)$ dla usunięć, wstawień i zamian znaków odpowiednio, to uogólniona odległość

Levenshteina między napisami `koza` i `foka` wynosi: $d_{\mathrm{lv}}(\texttt{koza}, \texttt{foka}) = 0.6$, bo $\texttt{koza} \xrightarrow[0.3]{zm.\ k\ na\ f} \texttt{foza} \xrightarrow[0.3]{zm.\ z\ na\ k} \texttt{foka}$.

Uogólniona odległość Levenshteina spełnia definicję metryki, gdy koszt usunięcia jest równy kosztowi wstawienia znaku. W przeciwnym przypadku nie spełnia ona założenia o symetrii. Jednakowoż, symetria zostaje zachowana przy jednoczesnej zamianie s i t oraz kosztów usunięcia i wstawienia znaku, jako że liczba usunięć znaków przy przetwarzaniu napisu s w napis t jest równa liczbie wstawień znaków przy transformacji napisu t w napis s [15]. Dobrze obrazuje to następujący przykład:

Przykład 1.7. Przyjmijmy za koszt usunięcia, wstawienia i zamiany znaku odpowiednio $(1, 0.1, 0.3)$. Wówczas uogólniona odległość Levenshteina dla napisów `koza` i `foczka` wynosi:

$$d_{\mathrm{lv}}(\texttt{koza}, \texttt{foczka}) = 0.5, \tag{1.5}$$

gdyż

$$\texttt{koza} \xrightarrow[0.3]{zm.\ k\ na\ f} \texttt{foza} \xrightarrow[0.1]{wst.c} \texttt{focza} \xrightarrow[0.1]{wst.k} \texttt{foczka},$$

natomiast

$$d_{\mathrm{lv}}(\texttt{foczka}, \texttt{koza}) = 2.3, \tag{1.6}$$

gdyż

$$\texttt{foczka} \xrightarrow[0.3]{zm.\ f\ na\ k} \texttt{koczka} \xrightarrow[1]{us.c} \texttt{kozka} \xrightarrow[1]{us.k} \texttt{koza}.$$

Gdy za koszty przyjmiemy $(1, 0.1, 0.3)$, to uogólniona odległość Levenshteina wynosi:

$$d_{\mathrm{lv}}(\texttt{koza}, \texttt{foczka}) = 2.3,$$

gdyż

$$\texttt{koza} \xrightarrow[0.3]{zm.\ k\ na\ f} \texttt{foza} \xrightarrow[1]{wst.c} \texttt{focza} \xrightarrow[1]{wst.\ k} \texttt{foczka},$$

czyli analogicznie, jak w przypadku 1.6. Natomiast

$$d_{\mathrm{lv}}(\texttt{foczka}, \texttt{koza}) = 0.5,$$

bo

$$\texttt{foczka} \xrightarrow[0.3]{zm.\ f\ na\ k} \texttt{koczka} \xrightarrow[0.1]{us.c|} kozka \xrightarrow[0.1]{us.\ k} \texttt{koza},$$

czyli analogicznie, jak w przypadku 1.5.

Zgodnie z lematem 1.5 nieścisła odległość Levenshteina jest równa ścisłej odległości Levenshteina. Z drugiej strony, ścisła odległość edycyjna jest równa kosztowi optymalnego dopasowania. Stąd rekurencja 1.2 liczy nieścisłą odległość Levenshteina. Częstym błędem jest, że następujące bezpośrednie uogólnienie, dodające transpozycję do zbioru bazowych operacji edycyjnych, rekurencji 1.4 liczy (nieścisłą) odległość Damerau-Levenshteina [4]:

$$C_{i,j} = \min \begin{cases} 0, & \text{gdy } i = j = 0 \\ C_{i-1,j} + 1, & \text{gdy } i > 0 \\ C_{i,j-1} + 1, & \text{gdy } j > 0 \\ C_{i-1,j-1} + [s_i \neq t_j], & \text{gdy } i, j > 0 \\ C_{i-2,j-2} + 1, & \text{gdy } s_i = t_{j-1}, s_{i-i} = t_j \text{ oraz } i, j > 1 \end{cases} \tag{1.7}$$

Jednak rekurencja 1.7 liczy odległość optymalnego dopasowania napisów, czyli ścisłą odległość Damerau-Levenshteina, która nie zawsze jest równa odległości Damerau-Levenshteina. Dla przykładu, odległość między napisami `ba` i `acb` wyliczona przy pomocy rekurencji 1.7 jest równa trzy, natomiast odległość Damerau-Levenshteina między tymi napisami wynosi dwa.

Rekurencyjna definicja odległości Damerau-Levenshteina została po raz pierwszy podana przez Lowrance'a i Wagnera [17]. W ich definicji zamiana zostaje zastąpiona poprzez minimalizację po możliwych transpozycjach między danym znakiem a wszystkimi nie przetransformowanymi znakami, przy czym koszt transpozycji wzrasta wraz z odległością między transponowanymi znakami [15]. Innymi słowy, do $\mathbb{B}$ należą wstawienia, usunięcia, zamiany oraz operacje $axb \to bya$ o koszcie równym $|x| + |y| + 1$ [4]. Mając tak zdefiniowane $\mathbb{B}$ ogólna rekurencja 1.2 dla odległości Damerau-Levenshteina przedstawia się następująco:

$$C_{i,j} = \min \begin{cases} 0, & \text{gdy } i = j = 0 \\ C_{i-1,j} + 1, & \text{gdy } i > 0 \\ C_{i,j-1} + 1, & \text{gdy } j > 0 \\ C_{i-1,j-1} + [s_i \neq t_j], & \text{gdy } i, j > 0 \\ \min\limits_{\substack{0<i'<i,\ 0<j'<j \\ s_i=t_{j'},\ s_{i'}=t_j}} C_{i'-1,j'-1} + (i - i') + (j - j') - 1 \end{cases} \tag{1.8}$$

Co więcej, Lowrance i Wagner wykazali, że wewnętrzne minimum w rekurencji 1.8 jest osiągane dla największych $i' < i$ oraz $j' < j$, które spełniają $s_i = t_{j'}$ oraz $s_{i'} = t_j$. Odległość Damerau-Levenshteina oznaczamy przez d_{dl}.

Przykład 1.8. Odległość optymalnego dopasowania napisów oraz Damerau-Levenshteina między napisami `kozak` i `foczka` wynosi 3, bo `kozak` $\xrightarrow[1]{zm.\ k\ na\ f}$ `fozak` $\xrightarrow[1]{wst.\ c}$ `foczak` $\xrightarrow[1]{transp.\ a\ i\ k}$ `foczka`.

W przypadku odległości Levenshteina, optymalnego dopasowania napisów oraz Damerau-Levenshteina, maksymalna odległość między napisami s i t wynosi $\max\{|s|, |t|\}$. Jednak warto zauważyć, że gdy liczba dopuszczalnych operacji edycyjnych rośnie, to liczba dopuszczalnych ścieżek między napisami wzrasta, co pozwala czasem zmniejszyć odległość między napisami. Dlatego relację między zaprezentowanymi powyżej odległościami można podsumować następująco [15]:

$$\left.\begin{array}{r} \infty(\geq |s|) \geq d_{\text{hamming}}(s,t) \\ |s| + |t| \geq d_{\text{lcs}}(s,t) \\ max\{|s|, |t|\} \end{array}\right\} \geq d_{\text{lv}}(s,t) \geq d_{osa}(s,t) \geq d_{dl}(s,t) \geq 0.$$

Jako że odległości Hamminga i najdłuższego wspólnego podnapisu nie mają wspólnych bazowych operacji edycyjnych, to nie ma pomiędzy nimi porządku relacyjnego. Górne ograniczenie $|s|$ odległości Hamminga jest zachowane jedynie gdy $|s| = |t|$.

1.2.2. Odległości oparte na q-gramach

Przypomnijmy, że q-gramem nazywamy napis składający się z q kolejnych (przylegających) znaków. q-gramy związane z napisem s są otrzymywane przez przesuwanie przez napis s

„okna" o szerokości q znaków i zapisaniu występujących q-gramów. Przykładowo digramy napisu `ela` to `el` i `la`. Oczywiście taka procedura nie ma sensu, gdy $q > |s|$ lub gdy $q = 0$. Z tego powodu definiujemy następujące przypadki brzegowe dla wszystkich odległości $d(s, t, q)$ opartych na q-gramach:

$$\begin{aligned} &d_q(s,t) = \infty, \text{ gdy } q > min\{|s|, |t|\}, \\ &d_0(s,t) = \infty, \text{ gdy } |s| + |t| > 0, \\ &d_0(\varepsilon, \varepsilon) = 0. \end{aligned}$$

Najprostszą odległością między napisami, opartą na q-gramach, otrzymuje się poprzez wypisanie unikalnych q-gramów w obu napisach i porównanie które są wspólne. Jeśli przez $\mathcal{Q}(s, q)$ oznaczymy zbiór unikalnych q-gramów występujących w napisie s, to możemy zdefiniować odległość Jaccarda [15]:

Definicja 1.7. *Niech $\mathcal{Q}(s, q)$ oznacza zbiór unikalnych q-gramów występujących w napisie s. Wówczas* odległość Jaccarda*, $d_{jaccard}$, między napisami s i t definiuje się jako*

$$d_{jaccard}(s, t, q) = 1 - \frac{|\mathcal{Q}(s, q) \cap \mathcal{Q}(t, q)|}{|\mathcal{Q}(s, q) \cup \mathcal{Q}(t, q)|},$$

gdzie $|\cdot|$ oznacza liczność zbioru.

Odległość Jaccarda przyjmuje wartości z przedziału $[0, 1]$, gdzie 0 odpowiada pełnemu pokryciu zbiorów, tj. $\mathcal{Q}(s, q) = \mathcal{Q}(t, q)$, natomiast 1 oznacza puste przecięcie, tj. $\mathcal{Q}(s, q) \cap \mathcal{Q}(t, q) = \emptyset$.

Przykład 1.9. Odległość Jaccarda między napisami `papaja` i `japa` dla $q = 2$ wynosi: $d_{jaccard}(\texttt{papaja}, \texttt{japa}, 2) = 0.25$, bo $\mathcal{Q}(\texttt{papaja}, 2) = \{\texttt{pa}, \texttt{ap}, \texttt{aj}, \texttt{ja}\}$, a $\mathcal{Q}(\texttt{japa}, 2) = \{\texttt{ja}, \texttt{ap}, \texttt{pa}\}$, więc odległość wynosi $1 - \frac{3}{4} = 0.25$.

Inną odległością opartą na q-gramach jest odległość q-gramowa [????]. Otrzymuje się ją poprzez wylistowanie q-gramów występujących w obu napisach i policzenie q-gramów, które nie są wspólne [dla obu napisów] [15]. Formalnie można to zapisać następująco:

Definicja 1.8. *Niech $s = s_1 s_2 \dots s_n$ będzie napisem z Σ^* i niech $x \in \Sigma^q$ będzie q-gramem. Jeśli $s_i s_{i+1} \dots s_{i+q-1} = x$ dla pewnego i, to x* wystąpiło *w s. Niech $\mathbf{v}(s, q)$ będzie wektorem o długości $|\Sigma|^q$, którego zmienne oznaczają liczbę wystąpień wszystkich możliwych q-gramów z Σ^q w s. Niech $s, t \in \Sigma^*$ oraz $q > 0$ będzie liczbą naturalną.* Odległość q-gramową *między napisami s i t definiuje się następująco [14]:*

$$d_{\texttt{q}gram}(s, t, q) = \|\mathbf{v}(s, q) - \mathbf{v}(t, q)\|_1 = \sum_{i=1}^{|\Sigma|^q} |v_i(s, q) - v_i(t, q)|. \tag{1.9}$$

Wzór 1.9 definiuje odległość q-gramową między napisami s i t jako odległość L_1 pomiędzy $\mathbf{v}(s, q)$ i $\mathbf{v}(t, q)$. Zauważmy, że, zamiast sprawdzać wystąpienie wszystkich możliwych q-gramów z Σ^q w napisach si t, wystarczy policzyć jedynie liczbę faktycznie występujących q-gramów w obu napisach, by obliczyć odległość q-gramową [15].

Przykład 1.10. Niech $\Sigma = \{\mathtt{a}, \mathtt{j}, \mathtt{p}\}$. Wówczas odległość q-gramowa między napisami `papaja` i `japa` dla $q = 2$ wynosi: $d_{\mathtt{q}gram}(\mathtt{papaja}, \mathtt{japa}, 2) = 2$. Wszystkie możliwe digramy występujące w napisach `papaja` i `japa` to `aj`, `ap`, `ja` i `pa`. Zatem $\mathbf{v}(\mathtt{papaja}, 2) = (1, 1, 1, 2)$, a $\mathbf{v}(\mathtt{japa}, 2) = (0, 1, 1, 1)$. Stąd $d_{\mathtt{q}gram}(\mathtt{papaja}, \mathtt{japa}, 2) = \|(1, 1, 1, 2) - (0, 1, 1, 1)\|_1 = 2$.

Maksymalna liczba wystąpień różnych q-gramów w napisie s wynosi $|s| - q - 1$. Stąd maksymalna odległość q-gramowa między napisami s i t wynosi $|s| + |t| - 2q - 2$, osiągana, gdy s i t nie mają wspólnych q-gramów [15].

Skoro zdefiniowana została odległość q-gramowa w języku wektorów, każda miara podobieństwa w (całkowitej) przestrzeni wektorowej może zostać zastosowana. Przykładowo można zdefiniować *odległość cosinusową* między napisami s i t:

$$d_{cos}(s, t, q) = 1 - \frac{\mathbf{v}(s, q) \cdot \mathbf{v}(t, q)}{\|\mathbf{v}(s, q)\|_2 \|\mathbf{v}ts, q)\|_2}, \tag{1.10}$$

gdzie $\|\cdot\|_2$ oznacza zwykłą normę Euklidesową. Odległość cosinusowa wynosi zero, gdy $s = t$ oraz jeden, gdy s i t nie mają wspólnych q-gramów. Odległość ta powinna być interpretowana jako kąt pomiędzy $\mathbf{v}(s, q)$ i $\mathbf{v}(t, q)$, jako że drugie wyrażenie równania 1.10 przedstawia cosinus kąta między dwoma wektorami.

Przykład 1.11. Niech $\Sigma = \{\mathtt{a}, \mathtt{j}, \mathtt{p}\}$. Wówczas odległość cosinusowa między napisami `papaja` i `japa` dla $q = 2$ wynosi: $d_{cos}(\mathtt{papaja}, \mathtt{japa}, 2) \approx 0.127$, bo $\mathbf{v}(\mathtt{papaja}, 2) = (1, 1, 1, 2)$, a $\mathbf{v}(\mathtt{japa}, 2) = (0, 1, 1, 1)$ (p. przykład 1.10), więc $d_{cos}(\mathtt{papaja}, \mathtt{japa}, 2) = 1 - \frac{4}{\sqrt{3} \cdot \sqrt{7}} \approx 0.127$.

Wszystkie trzy odległości oparte na q-gramach są nieujemne i symetryczne. Odległości Jaccarda i q-gramowa spełniają również nierówność trójkąta [DOWODY???], w odróżnieniu od odległości cosinusowej. Żadna z powyższych miar nie spełnia warunku identyczności, ponieważ zarówno $\mathcal{Q}(s, q)$, jak i $\mathbf{v}(s, q)$ jest funkcją wiele-do-jednego. Jako przykład, zauważmy, że $\mathcal{Q}(\mathtt{abaca}, 2) = \mathcal{Q}(\mathtt{acaba}, 2)$ oraz $\mathbf{v}(\mathtt{abaca}, 2) = \mathbf{v}(\mathtt{acaba}, 2)$, więc $d_{jaccard}(\mathtt{abaca}, \mathtt{acaba}, 2) = d_{\mathtt{q}gram}(\mathtt{abaca}, \mathtt{acaba}, 2) = d_{cos}(\mathtt{abaca}, \mathtt{acaba}, 2) = 0$. Innymi słowy, odległość oparta na q-gramach równa zero, nie gwarantuje, że $s = t$. [Inne własności $\mathbf{v}(s, q)$ można znaleźć w [14].]

1.2.3. Miary heurystyczne

Odległość Jaro została stworzona [wymyślona, zdefiniowana???] w amerykańskim Bureau of the Census (rządowa agencja, która jest odpowiedzialna m.in. za spis ludności Stanów Zjednoczonych) w celu połączenia rekordów, które były wpisane w niewłaściwe pola formularza oraz zlikwidowaniu literówek. Pierwszy publiczny opis tej odległości pojawił się w instrukcji obsługi citeJaro1978:usermanual, co może wyjaśniać dlaczego nie jest rozpowszechniona w literaturze informatycznej. Jednak odległość ta została skutecznie zastosowana w statystycznych problemach dopasowania w przypadku dość krótkich napisów, głównie imion, nazwisk oraz danych adresowych [15].

Rozumowanie stojące za odległością Jaro jest następujące: błędny znak oraz transpozycje znaków są spowodowane błędem przy wpisywaniu, ale mało prawdopodobne jest znalezienie błędnego znaku w miejscu odległym od zamierzonego, żeby mogło to być spowodowane

błędem przy wpisywaniu. Stąd odległość Jaro mierzy liczbę wspólnych znaków w dwóch napisach, które nie są zbyt odległe od siebie i dodaje karę za dopasowanie znaków, które są stransponowane. Formalna definicja wygląda następująco [15]:

Definicja 1.9. *Niech s i t będą napisami z Σ^*. Niech m oznacza liczbę wspólnych znaków z s i t, przy czym zakładając, że $s_i = t_j$, to znak ten jest* wspólny *dla obu napisów, jeśli:*

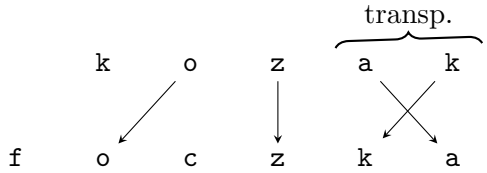
$$|i - j| < \left\lfloor \frac{max\{|s|, |t|\}}{2} \right\rfloor$$

i każdy znak z s może być wspólny ze znakiem z t tylko raz. W końcu, jeśli s' i t' są podnapisami utworzonymi z s i t poprzez usunięcie znaków, które nie są wspólne dla obu napisów, to T jest liczbą transpozycji potrzebnych to otrzymania t' z s'. Transpozycje znaków nieprzylegających są dozwolone.

Wówczas odległość Jaro *definiuje się jako:*

$$d_{jaro}(s,t) = \begin{cases} 0, & gdy\ s = t = \varepsilon \\ 1, & gdy\ m = 0\ i\ |s| + |t| > 0 \\ 1 - \frac{1}{3}(\frac{m}{|s|} + \frac{m}{|t|} + \frac{m-T}{m}) & w\ przeciwnym\ przypadku \end{cases} \tag{1.11}$$

Odległość Jaro przyjmuje wartości z przedziału $[0, 1]$, gdzie zero oznacza, że $s = t$, natomiast jeden wskazuje na kompletną odmienność napisów z $m = T = 0$.



Rysunek 1.4: Przykład odległości Jaro między napisami `kozak` i `foczka`.

Przykład 1.12. Odległość Jaro między napisami `kozak` i `foczka` wynosi: $d_{jaro}(\texttt{kozak}, \texttt{foczka}) \approx 0.261$, bo liczba wspólnych znaków wynosi $m = 4$, a liczba potrzebnych transpozycji wynosi $T = 1$ (p. rys. 1.4), co daje odległość równą $d_{jaro}(\texttt{kozak}, \texttt{foczka}) = 1 - \frac{1}{3}(\frac{3}{5} + \frac{4}{6} + \frac{3}{4}) = \frac{47}{180} \approx 0.261$.

Winkler rozszerzył odległość Jaro przez włączenie dodatkowej kary za błędny znak wśród pierwszych czterech znaków napisu [15]:

Definicja 1.10. *Niech s i t będą napisami z Σ^*, $\ell(s,t)$ oznacza długość najdłuższego wspólnego prefiksu, mającego maksymalnie cztery znaki i niech p będzie liczbą z przedziału $[0, \frac{1}{4}]$. Wówczas odległość Jaro-Winklera dana jest wzorem [18]:*

$$d_{jw}(s,t,p) = d_{jaro}(s,t)[1 - p\ell(s,t)] \tag{1.12}$$

Czynnik p określa jak bardzo różnice w czterech pierwszych znakach w obu napisach wpływają na odległość między nimi. Zmienna p jest liczbą z przedziału $[0, \frac{1}{4}]$, by mieć pewność,

że odległość Jaro-Winklera miała wartości w przedziale $[0, 1]$ ($0 \leq d_{jw}(s, t) \leq 1$). Jeśli $p = 0$, to odległość ta redukuje się do odległości Jaro i wszystkie znaki wnoszą taki sam wkład do funkcji odległości. Jeśli $p = \frac{1}{4}$, to odległość Jaro-Winklera jest równa zero nawet wówczas gdy tylko cztery pierwsze znaki w obu napisach pokrywają się. Powód jest taki, że podobno ludzie są mniej skłonni do popełniania błędów w czterech pierwszych znakach lub też są one lepiej zauważalne, więc różnice w pierwszych czterech znakach wskazują na większe prawdopodobieństwo, że dwa napisy są rzeczywiście różne [15]. Wikler [18] używał w swoich badaniach $p = 0.1$ i zauważył lepsze rezultaty niż dla $p = 0$.

Przykład 1.13. Odległość Jaro-Winklera między napisami `faktura` i `faktyczny` dla $p = 0$, $p = 0.1$ oraz $p = 0.25$ wynosi odpowiednio:

$$d_{jw}(\texttt{faktura}, \texttt{faktyczny}, p = 0.00) \approx 0.328 = d_{jaro}(\texttt{faktura}, \texttt{faktyczny})$$
$$d_{jw}(\texttt{faktura}, \texttt{faktyczny}, p = 0.10) \approx 0.197$$
$$d_{jw}(\texttt{faktura}, \texttt{faktyczny}, p = 0.25) = 0$$

Łatwo zauważyć z równań 1.11 i 1.12, że odległości Jaro i Jaro-Winklera, dla $p \neq \frac{a}{4}$, są nieujemne, symetryczne i spełniają warunek identycznościowy [DOWOD??]. Nierówność trójkatą w obu przypadkach nie jest jednak spełniona. Rozważmy następujący przykład: $s = \texttt{ab}, t = \texttt{cb}, u = \texttt{cd}$. Jako że napisy s i u nie mają wspólnych znaków, to odległość Jaro między nimi wynosi $d_{jaro}(s, u) = 0$, podczas gdy $d_{jaro}(s, t) = d_{jaro}(t, u) = \frac{1}{3}$, więc w tym przypadku $d_{jaro}(s, u)$ jest większe od $d_{jaro}(s, t) + d_{jaro}(t, u)$. Z tego łatwo zauważyć, że odległość Jaro-Winklera nie spełnia nierówności trójkąta dla tego samego przykładu dla $p \in [0, \frac{1}{4}]$ [15].

W niniejszym rozdziale przedstawiono odległości określone na napisach [czy też przestrzeni ciągów znaków]. Mając do wyboru wachlarz różnych funkcji nasuwa się pytanie której użyć. Ostateczna decyzja zależy od konkretnego przypadku, jednak istnieją pewne ogólne reguły. Wybór pomiędzy odległościami opartymi na operacjach edycyjnych i q-gramach z jednej strony, a miarami heurystycznymi z drugiej zależy w dużej mierze od długości napisów – te ostatnie są dedykowane krótszym napisom takim jak np. dane osobowe. W odróżnieniu od odległości opartych na operacjach edycyjnych i miarach heurystycznych, odległości oparte na q-gramach można łatwo policzyć dla bardzo długich tekstów, jako że liczba q-gramów możliwych do utworzenia z języka naturalnego (dla niezbyt małego q, tj. $q \geq 3$) jest z reguły o wiele mniejsza niż liczba q-gramów, którą można otrzymać z całego alfabetu. Wybór pośród odległości opartych na operacjach edycyjnych zależy przede wszystkim od dokładności jaką chce się otrzymać. Przykładowo do wyszukiwania haseł w słowniku, gdzie różnice między dobranymi napisami są niewielkie, odległości pozwalające na więcej operacji edycyjnych (tak jak np. odległość Damerau-Levenshteina) mogą dać lepsze rezultaty. Odległości Jaro i Jaro-Winklera zostały skonstruowane do krótkich, napisanych przez człowieka, napisów, więc ich zakres zastosowania powinien być jasny.

Rozdział 2

Analiza skupień metodą k-średnich

Analiza skupień polega na wyróżnieniu w zbiorze ustalonej liczby rozłącznych skupień obserwacji w jakimś sensie do siebie podobnych, równocześnie zachowując maksymalne zróżnicowanie obserwacji pomiędzy poszczególnymi podzbiorami [7]. W niniejszym rozdziale przedstawimy analizę skupień metodą k-średnich w trzech odsłonach: przedstawimy metodę wsadową, przy użyciu stochastycznego spadku gradientu oraz metodę pośrednią, tzw. miniwsadową.

2.1. Metoda k-średnich

Rozważmy przestrzeń euklidesową $\mathbb{R}^p$ i niech będzie dana liczba skupień k. Wówczas zadanie znalezienia skupień o wyżej wymienionych własnościach można sprowadzić do dobrze określonego zadania optymalizacji. Weźmy próbę n-elementową obserwacji $\mathbf{x}_i,\ i = 1, \ldots, n$ o wartościach w $\mathbb{R}^p$. Suma kwadratów odległości między obserwacjami próby wynosi [7]

$$T = \frac{1}{2} \sum_{i}^{n} \sum_{j}^{n} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \tag{2.1}$$

gdzie $\|\cdot\|_2$ oznacza normę euklidesową. Niech funkcja $C : \{1, \ldots, n\} \to \{1, \ldots, k\}$ oznacza przydzielenie danej obserwacji danemu skupieniu, tzn. jeśli $C(i) = l$, to oznacza, że $\mathbf{x}_i$ należy do l-tego skupienia. Zakładając, że dokonano podziału próby na k podzbiorów, można całkowitą sumę kwadratów rozłożyć na sumę kwadratów odległości między obserwacjami z tego samego skupienia oraz na sumę kwadratów odległości między obserwacjami z różnych skupień [7]:

$$T = W + B = \frac{1}{2} \sum_{l=1}^{k} \sum_{C(i)=k} \sum_{C(j)=k} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 + \frac{1}{2} \sum_{l=1}^{k} \sum_{C(i)=k} \sum_{C(j)\neq k} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \tag{2.2}$$

Mając tak sformułowany rozkład sumy T widzimy, że zmieniając podział punktów na skupienia zmienia się zarówno suma W, jak i B. Można więc sformułować problem analizy skupień jako zadanie minimalizacji sumy W lub, równoważnie, maksymalizacji sumy B. Maksymalizacja B to po prostu maksymalizacja rozproszenia punktów z różnych podzbiorów, co jest

równoznaczne z minimalizacją rozproszenia punktów z tego samego skupienia. Stąd, rozwiązaniem problemu analizy skupień jest dokonanie takiego podziału próby, aby zminimalizować sumę W. Ze względu na złożoność obliczeniową, niemożliwe jest bezpośrednie rozwiązanie tego problemu [7].

Przez n_l oznaczmy liczność l-tego skupienia i niech $\mathbf{m}_l = \frac{1}{n_l} \sum_{C(i)=k} \mathbf{x}_i$ oznacza wektorową średnią obserwacji z l-tego skupienia. Łatwo zauważyć, że [7]

$$W = \frac{1}{2} \sum_{l=1}^{k} \sum_{C(i)=l} n_l \|\mathbf{x}_i - \mathbf{m}_l\|_2^2 \tag{2.3}$$

Średnie $\mathbf{m}_l$, $l = 1, \ldots, k$ nazywamy *środkami skupień*. Równanie 2.3 można uprościć do następującej postaci, która w praktyce jest łatwa w optymalizacji:

$$\widetilde{W} = \frac{1}{2} \sum_{l=1}^{k} \sum_{C(i)=l} \|\mathbf{x}_i - \mathbf{m}_l\|_2^2 = \frac{1}{2} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{m}_{C(i)}\|_2^2 \tag{2.4}$$

Algorytmy, które rozwiązują problem minimalizacji sumy 2.4, znane są pod nazwą *metody k-średnich*.

2.2. Algorytmy

2.2.1. Algorytm wsadowy

Algorytm *wsadowy* (ang. *batch algorithm*) zostaje zainicjalizowany przez losowe wyznaczenie k punktów jako początkowe środki skupień. Dalej następuje przydzielenie wszystkich punktów próby do najbliższego skupienia, a następnie przeliczenie środków jako średniej ze wszystkich obserwacji w danym skupieniu. Procedura ta jest powtarzana aż do ustabilizowania się algorytmu, tj. do momentu aż żaden punkt próby nie zmieni skupienia [19].

Algorithm 1 Algorytm wsadowy k-średnich

given: k, data set X
initialize randomly m_l, $\forall l = 1, ..., k$
repeat
    for $i = 1, \ldots, n$ **do**
        $C(i) = \arg\min_l \|\mathbf{x}_i - \mathbf{m}_l\|_2^2$
    end for
    for $l = 1, \ldots, k$ **do**
        $\mathbf{m}_l = \frac{1}{n_l} \sum_{C(i)=l} \mathbf{x}_i$
    end for
until convergence

Algorytm wsadowy jest najbardziej popularnym i najczęściej stosowanym algorytmem, gdyż jest szybki i zazwyczaj daje dobre rezultaty. Jednakowoż jeśli liczba obserwacji w zbiorze jest bardzo duża, to obliczanie średnich z obserwacji we wszystkich skupieniach jest bardzo kosztowne obliczeniowo, zbiegając w czasie $O(knp)$, gdzie p to liczba zmiennych. Stąd Bottou i Bengio [3] zaproponowali algorytm oparty na stochastycznym spadku gradientu.

2.2.2. Algorytmy oparte na spadku gradientu

Algorytmy oparte na spadku gradientu są często stosowane np. w regresji liniowej [2]. Idea polega na szukaniu minimum z danej funkcji kosztu, w kolejnych krokach algorytmu aktualizując zmienną, w kierunku, w którym spadek gradientu był największy. Każda aktualizacja zależy od parametru, zwanego *parametrem uczenia*, który musi być odpowiednio dobrany. W niniejszej sekcji [podrozdziale?] opiszemy trzy algorytmy oparte na spadku gradientu.

Mając daną funkcję kosztu $\widetilde{W} = \widetilde{W}(\mathbf{m}, \mathbf{x}_i) = \frac{1}{2}\sum_{i=1}^{n}\|\mathbf{x}_i - \mathbf{m}_{C(i)}\|_2^2$, możemy znaleźć minimum używając tzw. *spadku gradientu*. W każdej iteracji algorytmu uaktualniamy wektor $\mathbf{m}$ na podstawie gradientu $\widetilde{W}(\mathbf{m}, \mathbf{x}_i)$:

$$\mathbf{m}_l^{(t+1)} = \mathbf{m}_l^{(t)} + \gamma \sum_{i=1}^{n} \frac{\partial \widetilde{W}(\mathbf{m}, \mathbf{x}_i)}{\partial \mathbf{m}} \tag{2.5}$$

gdzie γ jest odpowiednio dobranym *parametrem uczenia*[, a t oznacza iterację algorytmu - PISAC TO??] [2]. Parametrem uczenia, które daje najlepsze rezultaty dla algorytmu k-średnich jest $\frac{1}{n_{C(i)}}$. Stąd też algorytm *wsadowego spadku gradientu*, w każdej iteracji algorytmu aktualizuje wektor $\mathbf{m}$ następująco [3]:

$$\mathbf{m}_l^{(t+1)} = \mathbf{m}_l^{(t)} + \sum_{C(i)=l} \frac{1}{n_l}(\mathbf{x}_i - \mathbf{m}_l^{(t)}) \tag{2.6}$$

Algorytm *stochastycznego spadku gradientu*, ozn. SGD, jest daleko idącym uproszczeniem. Zamiast liczyć gradient z $\widetilde{W}(\mathbf{m}, \mathbf{x}_i)$ wprost, każda iteracja estymuje gradient na podstawie *jednej losowo wybranej* obserwacji $\mathbf{x}_i$ [2]:

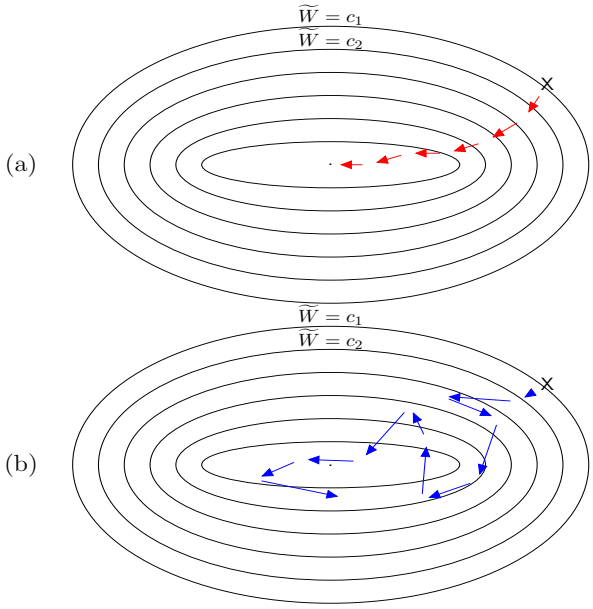
$$\mathbf{m}_l^{(t+1)} = \mathbf{m}_l^{(t)} + \gamma \frac{\partial \widetilde{W}(\mathbf{m}, \mathbf{x}_i)}{\partial \mathbf{m}} \tag{2.7}$$

gdzie γ jest odpowiednio dobranym parametrem uczenia. Tak samo jak w przypadku algorytmu wsadowego, parametrem uczenia, które daje najlepsze rezultaty jest $\frac{1}{n_{C(i)}}$. Stąd też algorytm stochastycznego spadku gradientu w każdej iteracji algorytmu aktualizuje wektor $\mathbf{m}$ następująco [3]:

$$n_l^{(t+1)} = n_l^{(t)} + \begin{cases} 1, & \text{gdy } l = C(j) \\ 0, & \text{wpp.} \end{cases} \tag{2.8}$$

$$\mathbf{m}_l^{(t+1)} = \mathbf{m}_l^{(t)} + \begin{cases} \frac{1}{n_l}(\mathbf{x}_i - \mathbf{m}_l^{(t)}), & \text{gdy } l = C(i) \\ 0, & \text{wpp.} \end{cases} \tag{2.9}$$

Algorytm SGD opiera się na przeliczaniu średniej po każdym przydzieleniu obserwacji do skupienia, choć z powodu stochastycznego szumu, takie rozwiązanie może nie prowadzić do lokalnego minimum, a jedynie w jego „pobliże". Na rys. 2.1 przedstawiono przykładową drogę aktualizacji parametrów. Kolejne elipsy oznaczają stałą wartość funkcji kosztu $\widetilde{W}(\mathbf{m}, \mathbf{x}_i)$ w zależności od wartości zmiennej $\mathbf{m}$, a centrum (środek?) oznacza (lokalne) minimum tej funkcji. Jeśli algorytm rozpoczyna działanie w punkcie oznaczonym przez X, to w przypadku algorytmu wsadowego spadku gradientu, w kolejnych iteracjach zmienna $\mathbf{m}$ zmienia swoją



Rysunek 2.1: Przykładowa droga wsadowego spadku gradientu (rys. a) i stochastycznego spadku gradientu (rys. b).

wartość, przybliżając się do (lokalnego) minimum funkcji $\widetilde{W}$. Natomiast algorytm stochastycznego spadku gradientu w każdej iteracji przybliża się w stronę minimum w sposób losowy, tzn. może nigdy nie osiągnąć właściwego minimum, a jedynie „krążyć" wokół niego.

Algorytm oparty na stochastycznym spadku gradientu (znany również pod nazwą algorytmu *online'owego* [I TAK I NIE - ZALEZY OD ARTICLE'A]) rozpoczyna się tak samo, tj. inicjalizacją losowych k środków skupień. Następnie zbiór obserwacji jest mieszany i obserwacje po kolei są przydzielane do najbliższego skupienia. Środki skupień przeliczane są po każdym przydzieleniu punktu do skupienia. Procedura ta powtarzana jest t razy [3]. Algorytm ten jest dużo szybszy od dwóch wcześniejszych, kosztem dokładności rozwiązania [2].

Algorithm 2 Algorytm SGD k-średnich

given: k, data set X, iterations t
initialize randomly $m_l,\ \forall l = 1, ..., k$
initialize $n_l = 0,\ \forall l = 1, ..., k$
for $a = 1, \ldots, t$ **do**
    randomly pick one observation $\mathbf{x}_i$ from X
    $C(i) = \arg\min_l \|\mathbf{x}_i - \mathbf{m}_l\|_2^2$
    $n_{C(i)} = n_{C(i)} + 1$
    $\mathbf{m}_{C(i)} = \mathbf{m}_{C(i)} + \frac{1}{n_{C(i)}} \|\mathbf{x}_i - \mathbf{m}_{C(i)}\|_2$
end for

Algorytm *mini-wsadowy* (ang. *mini-batch k-means*) jest połączeniem dwóch poprzednich algorytmów, tj. w każdej iteracji przydzielanych do najbliższego skupienia jest b losowo wybranych obserwacji, po czym następuje przeliczenie środków skupień [12]. Algorytm ten jest porównywalnie szybki do algorytmu SGD, osiągając przy tym lepsze rezultaty z powodu mniejszego stochastycznego szumu.

Algorithm 3 Algorytm mini-wsadowy k-średnich

given: k, data set X, iterations t, mini-batch size b
initialize randomly $m_l,\ \forall l = 1, ..., k$
initialize $n_l = 0,\ \forall l = 1, ..., k$
for $a = 1, \ldots, t$ **do**
    $B = b$ observations randomly picked from X
    for $i : \mathbf{x}_i \in B$ **do**
        $C(i) = \arg\min_l \|\mathbf{x}_i - \mathbf{m}_l\|_2^2$
    end for
    for $i : \mathbf{x}_i \in B$ **do**
        $n_{C(i)} = n_{C(i)} + 1$
        $\mathbf{m}_{C(i)} = \mathbf{m}_{C(i)} + \frac{1}{n_{C(i)}} \|\mathbf{x}_i - \mathbf{m}_{C(i)}\|_2$
    end for
end for

[CZY PODAWAC TUTAJ PRZYKLAD + RYSUNKI Z [12] O TYM ZE MINI-BATCH JEST TAKI SUPER W POR. Z INNYMI??]

2.3. Metody oceny jakości podziału na skupienia

[TUTUAJ KORZYSTALAM MOCNO Z DOKUMENTACJI SCIKIT: `http://scikit-learn.org/stable/modules/clustering.html#clustering-evaluation` ZAWRZEC TO GDZIES??]

Ocena skuteczności działania algorytmów analizy skupień nie należy do zadań tak prostych jak np. ocena modelu klasyfikacji pod nadzorem. W szczególności żadna metoda ocena nie powinna brać pod uwagę wartości etykiety skupienia, ale powinna sprawdzać, czy zbiór danych jest dobrze podzielony, tzn. czy obserwacje w poszczególnych skupieniach są do siebie „podobne", a obserwacje z różnych skupień – „niepodobne", zgodnie z przyjętą metryką podobieństwa. W niniejszych podrozdziale przedstawimy kilka miar oceny jakości podziału na skupienia.

Przyjmujemy następującego założenia: Niech K i C oznaczają dwa różne podziały n-elementowego zbioru X na skupienia. Zazwyczaj K oznacza podział uzyskany przy pomocy algorytmu dzielącego zbiór, a C jest zbiorem prawdziwych klas, do których należą obserwacje, choć K i C mogą również oznaczać dwa niezależne podziały uzyskane przy pomocy różnych algorytmów.

2.3.1. Skorygowany indeks Randa []Adjusted Rand Index [JAK TO PRZETŁUMACZYĆ ???]]

Niech a_1 oznacza liczbę par elementów z X, które mają wspólne skupienie zarówno w K, jak i w C, natomiast przez a_0 oznaczmy liczbę par elementów z X, które mają zostały przypisane do innych skupień zarówno w K, jak i w C. Elementy, które spełniają jeden z powyższych warunki oznaczają zgodność podziałów K i C. Możemy wówczas zdefiniować *indeks Randa* [9]:

$$\mathrm{RI} = \frac{a_0 + a_1}{\binom{n}{2}} \tag{2.10}$$

Wartości indeksu Randa znajdują się w przedziale $[0, 1]$. W praktyce jednak RI leży często pomiędzy 0.5, a 1. Co więcej, miara ta nie gwarantuje, że losowy podział zbioru da wartość indeksu bliską zeru. Z tego powodu RI jest zazwyczaj używana w skorygowanej formie [6]:

$$\mathrm{ARI} = \frac{2(a_0 a_1 - b_0 b_1)}{(a_0 + b_0)(b_0 + a_1) + (a_0 + b_1)(b_1 + a_1)}, \tag{2.11}$$

gdzie b_0 oznacza liczbę par elementów z X, które należą do tego skupienia w K, ale do różnych skupień w C, natomiast b_1 oznacza liczbę par elementów z X, które należą do różnych skupień w K, ale do tego samego skupienia w C.

Zalety:

- Losowe przyporządkowanie do skupień daje wartość ARI bliską zeru dla dowolnej liczby skupień i obserwacji w zbiorze.
- Miara ARI daje wartości z przedziału $[-1, 1]$, gdzie -1 oznacza niezależne przyporządkowanie, natomiast 1 oznacza pełną zgodność.
- Brak założeń o strukturze skupienia: przy pomocy ARI można porównywać podział na skupienia uzyskany przy pomocy różnych algorytmów, które mają różne założenia o strukturze skupienia.

Wady:

- W przypadku sprawdzenia jakości działania jednego algorytmu, wymagana jest znajomość prawdziwego podziału zbioru, co w praktyce rzadko występuje lub wymaga ręcznego podziału zbioru.

2.3.2. Jednorodność, zupełność oraz miara V

Niech C oznacza zbiór prawdziwych klas, do których należą obserwacje. Mówimy, że podział zbioru jest *jednorodny*, jeśli wszystkie skupienia zawierają jedynie obserwacje z jednej klasy. Podział zbioru jest *zupełny*, jeśli wszystkie obserwacje z danej klasy są w tym samym skupieniu. Jednorodność i zupełność podziału może być często w opozycji do siebie, tzn. gdy jednorodność rośnie, to zupełność zazwyczaj maleje i odwrotnie. Przykładowo, rozważmy dwa skrajne podziały. W przypadku pierwszego, gdy przydzielamy wszystkie obserwacje do jednego skupienia, to dostajemy idealną zupełność – wszystkie elementy z jednej klasy należą do tego samego skupienia. Jednakowoż, podział taki jest tak *nie*jednorodny, jak to tylko możliwe, skoro wszystkie klasy znajdują się w jednym skupieniu. Z drugiej strony, rozważmy przydzielenie każdej obserwacji do osobnego skupienia. W tym przypadku, podział jest idealnie jednorodny – każde skupienie zawiera jedynie obserwacje z jednej klasy. Jednak w terminach zupełności, taki podział bardzo słaby, chyba że rzeczywiście każda klasa zawiera jeden element. Miara V jest ważoną średnią harmoniczną dwóch powyższych miar [10].

Jednorodność. Żeby spełnić kryteria, jak musi spełniać podział jednorodny, każde ze skupień musi zawierać tylko i wyłącznie te obserwacje, które należą do jednej klasy. To znaczy, że rozkład klas w każdym ze skupień powinien skośny i zawierać tylko jedną klasę, tj. entropia powinna wynosić zero. Aby ustalić jak blisko od idealnego znajduje się dany podział, sprawdzamy warunkową entropię rozkładu klas pod warunkiem zaproponowanego podziału. W idealnie jednorodnym podziale, tak wartość, tj. $H(C|K)$ wynosi zero. Jednak w nieidealnej

sytuacji, wartość ta zależy od wielkości zbioru i rozkładu klas. Stąd, zamiast badać warunkową entropię, normalizujemy tę wartość przez maksymalną redukcję entropii jaką informacja o podziale może przynieść, tj. $H(C)$ [10].

Zauważmy, że $H(C|K)$ jest maksymalne (i równe $H(C)$), kiedy podział na skupienia nie wnosi żadnej nowej informacji – rozkład klas w każdym skupieniu jest równy ogólnemu rozkładowi klas. $H(C|K)$ jest równy zeru, gdy każde skupienie zawiera jedynie obserwacje z jednej klasy, tj. w przypadku idealnie jednorodnego podziału. W przypadku zdegenerowanym, gdy $H(C) = 0$, kiedy istnieje tylko jedna klasa, definiujemy jednorodność jako równą 1. W idealnie jednorodnym rozwiązaniu, taka normalizacja, tj. $\frac{H(C|K)}{H(C)}$, wynosi zero. Stąd, aby utrzymać konwencję, że wartość 1 jest pożądana, a wartość 0 jest niepożądana, definiujemy jednorodność jako [10]:

$$h = \begin{cases} 1, & \text{gdy } H(C,K) = 0 \\ 1 - \frac{H(C|K)}{H(C)}, & \text{w przeciwnym przypadku} \end{cases} \tag{2.12}$$

gdzie

$$H(C) = -\sum_{c=1}^{|C|} \frac{n_c}{n} \cdot \log \frac{n_c}{n},$$
$$H(C|K) = -\sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \cdot \log \frac{n_{c,k}}{n_k},$$

gdzie n_c oznacza liczność klasy c, $c = \{1, \ldots, |C|\}$, a $n_{c,k}$ oznacza liczbę elementów, która należy do klasy c i skupienia k, $k = \{1, \ldots, |K|\}$.

Zgodność. Miara zgodności jest symetryczna do miary jednorodności. Aby spełnić warunki zgodności, podział zbioru musi przydzielić wszystkie obserwacje z jednej klasy, do tego samego skupienia. Żeby policzyć zgodność, badamy rozkład przypisanych skupień w obrębie jednej klasy. W idealnie zgodnym podziale, każdy z rozkładów będzie skośny i będzie zawierać jedynie tylko jedno skupienie. Możemy oszacować poziom skośności, licząc warunkową entropię zaproponowanego podziału pod warunkiem klas, tj. $H(K|C)$. W idealnie zgodnym rozwiązaniu $H(K|C) = 0$. Jednak w najgorszym możliwym przypadku, gdy wszystkie obserwacje z tej samej klasy są we wszystkich skupieniach, rozkład tych ostatnich jest równy rozkładowi rozmiarów klastrów, $H(K|C)$ jest maksymalne i równe $H(K)$. W końcu, w zdegenerowanym przypadku, kiedy $H(K) = 0$, kiedy mamy tylko jedno skupienie, definiujemy zgodność jako równą 1. Stąd, robiąc symetryczne wyliczenie do poprzedniego, definiujemy zgodność jako [10]:

$$c = \begin{cases} 1, & \text{gdy } H(K,C) = 0 \\ 1 - \frac{H(K|C)}{H(K)}, & \text{w przeciwnym przypadku} \end{cases} \tag{2.13}$$

gdzie

$$H(K) = -\sum_{k=1}^{|K|} \frac{n_k}{n} \cdot \log \frac{n_k}{n},$$
$$H(K|C) = -\sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \cdot \log \frac{n_{c,k}}{n_c}$$

Miara V. Mając zdefiniowane miary jednorodności i zgodności, możemy wyliczyć miarę V jako ważoną średnią harmoniczną tych dwóch miar [10]:

$$\mathrm{V}_\beta = \frac{(1+\beta)\cdot h \cdot c}{(\beta \cdot h) + c} \tag{2.14}$$

Zauważmy, że jeśli β jest mniejsza niż 1, jednorodność ma większą wagę niż zgodność. Często za β przyjmuje się po prostu 1, dając równą wagę obu miarom.

Warto zwrócić uwagę na fakt, że jednorodność, zgodność oraz miara V są niezależne od liczby klas, skupień, liczby obserwacji oraz użytego algorytmu. Stąd miary te mogą być używane do porównania każdego algorytmu dzielącego na skupienia, niezależnie od powyższych parametrów. Co więcej, wyliczając zarówno jednorodność, jak i zgodność, może zostać otrzymana bardziej precyzyjna ocena jakości podziału skupień.

Zalety:

- Miary te dają wartości z przedziału $[0, 1]$, gdzie 0 oznacza najgorszy możliwy przypadek, natomiast 1 to bardzo dobre rozwiązanie.
- Intuicyjna interpretacja: podział z niską wartością miary V może zostać oceniony w terminach jednorodności i zgodności, aby mieć lepsze pojęcie o błędach jakich dokonał algorytm.
- Brak założeń o strukturze skupienia: przy pomocy powyższych miar można porównywać podział na skupienia uzyskany przy pomocy różnych algorytmów, które mają różne założenia o strukturze skupienia.

Wady:

- Powyższe miary nie są znormalizowane pod względem losowego przypisania do skupienia. Oznacza to, że w zależności od liczby obserwacji, podziału na skupienia i klasy, losowy przydział do skupień nie zawsze da takie same wartości jednorodności, zgodności oraz miary V. W szczególności, losowe przyporządkowanie może nie dać wartości powyższych miar równych zero, zwłaszcza gdy liczba skupień jest duża. Problem ten może być bezpiecznie zignorowany, gdy liczba obserwacji jest większa od tysiąca, a liczba skupień mniejsza niż 10. Dla mniejszej próbki i większej liczby skupień, bezpieczniej jest używać miary ARI. [CHYBA TEGO NIE ROZUEMIM - W ZALETACH BYLO, ZE MIARY TE SA NZAL. OD LICZNOSCI PROBKI/KLAS/KLASTROW...]
- W przypadku sprawdzenia jakości działania jednego algorytmu, wymagana jest znajomość prawdziwego podziału zbioru, co w praktyce rzadko występuje lub wymaga ręcznego podziału zbioru.

2.3.3. Miara silhouette

Sylwetki obserwacji (czy też *silhouettes*) są użyteczne, gdy odległości są określone na relatywnej skali (ratio scale??) oraz gdy pożądane są wyraźnie odseparowane skupienia. Aby skonstruować sylwetkę obserwacji potrzebne są dwie rzeczy: podział zbioru na skupienia C oraz miarę odległości d pomiędzy obserwacjami [11].

Weźmy każdą obserwację z próbki $\mathbf{x}_i$ i oznaczmy przez $C(i)$ skupienie, do którego należy. Średnią odmiennością $\mathbf{x}_i$ od swojego skupienia nazywamy [11]:

$$a(\mathbf{x}_i) = \frac{\sum_{u \in C(i)} d(\mathbf{x}_i, u)}{n_{C(i)}}$$

Średnią odmiennością $\mathbf{x}_i$ od skupienia J nazywamy:

$$c(\mathbf{x}_i, J) = \frac{\sum_{u \in J} d(\mathbf{x}_i, u)}{n_J}$$

Wówczas możemy wyliczyć odległość obserwacji $\mathbf{x}_i$ od najbliższego skupienia innego niż $C(i)$:

$$b(\mathbf{x}_i) = \min_{j \neq i} c(\mathbf{x}_i, J)$$

Skupienie L, dla którego minimum jest osiągnięte (tj. $b(\mathbf{x}_i) = c(\mathbf{x}_i, L)$) nazywamy sąsiadem obserwacji $\mathbf{x}_i$. Jest ono drugim najlepszym wyborem dla tej obserwacji. Stąd, znajomość sąsiada jest bardzo użyteczna, gdyż mówi o tym które skupienie zostałoby wybrane, gdyby nie zostało nim skupienie $C(i)$. Sylwetka obserwacji jest definiowana następująco [11]:

$$s(\mathbf{x}_i) = \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{\max b(\mathbf{x}_i), a(\mathbf{x}_i)}$$

Sylwetki otrzymujemy dla każdej obserwacji z osobna, ale najczęściej interesująca jest miara zdefiniowana dla całego zbioru, stąd sylwetką zbioru nazywamy średnią z sylwetek po wszystkich obserwacjach:

$$sil = \sum_{i=1}^{n} s(\mathbf{x}_i)$$

Zalety:

- Miara ta daje wartości z przedziału $[-1, 1]$, gdzie -1 oznacza niepoprawny podział, natomiast 1 to bardzo gęste skupienia. Wartości w okolicy zera sugerują nakładające się skupienia.
- Wartość sylwetki jest wyższa, gdy skupienia są gęste i dobrze odseparowane, co jest jedną z najbardziej pożądanych cech analizy skupień.
- Miara nie wymaga znajomości prawdziwych klas, na jakie można podzielić zbiór.

Wady:

- Miara silhouette przyjmuje w ogólności większe wartości dla wypukłych skupień.

2.4. Metody hierarchiczne

[TUTAJ CZY WCZESNIEJ??] Opis metody hierarchicznej, z wyróżnieniem odmienności najbliższego, najdalszego sąsiada oraz średniej.

Rozdział 3

Kategoryzacja tematyczna tekstów przy użyciu metryk w przestrzeni ciągów znaków

Wikipedia[1] jest to wielojęzyczna encyklopedia internetowa, która działa w oparciu o zasadę otwartej treści. Portal umożliwia każdemu z użytkowników odwiedzających stronę, edycję i aktualizację treści w czasie rzeczywistym. Wikipedia ma ponad 35.9 miliona artykułów we wszystkich wersjach językowych, w tym prawie 5 milionów w wersji angielskiej i nieco ponad 1.1 miliona artykułów w języku polskim (dane na sierpień 2015) [1].

Każdy artykuł może być edytowany przez dowolnego użytkownika, jak również nowe teksty może stworzyć każda osoba odwiedzająca portal. Przy procesie edycji oraz pisania artykułu obowiązują liczne reguły, między innymi takie jak wskazanie źródeł bibliograficznych, podlinkowanie do innych artykułów oraz nadanie artykułowi kategorii tematycznych [1].

Ta ostatnia zasada może w szczególności przysporzyć nieco kłopotów. Liczba dostępnych kategorii jest bardzo duża (ponad 125 tys. w polskiej wersji językowej), co więcej poziom ich szczegółowości jest zróżnicowany, tzn. mamy kategorie bardzo ogólne (np. *Matematyka*), jak i dość szczegółowe (np. *Działania dwuargumentowe*). Można się spodziewać, że automatyczny podział tekstów na kategorie na podstawie słów, jakie w nich występują, mógłby dać lepszy, bardziej dopasowany do treści, temat. Sprawdzenie jak automatyczny podział tekstów na kategorie tematyczne, przy użyciu występujących w nich słów oraz ich liczności, jest zasadniczym celem tej pracy. Idea działania jest następująca: algorytm analizuje jakie słowa w jakich ilościach występują w danym artykule i następnie przydziela go do grupy artykułów które zawierają takie same i podobne słowa w zbliżonych licznościach. To podejście opiera się na założeniu, że artykuły o podobnej tematyce będą zawierały takie same lub podobne do siebie słowa.

Schemat działania algorytmu jest następujący:

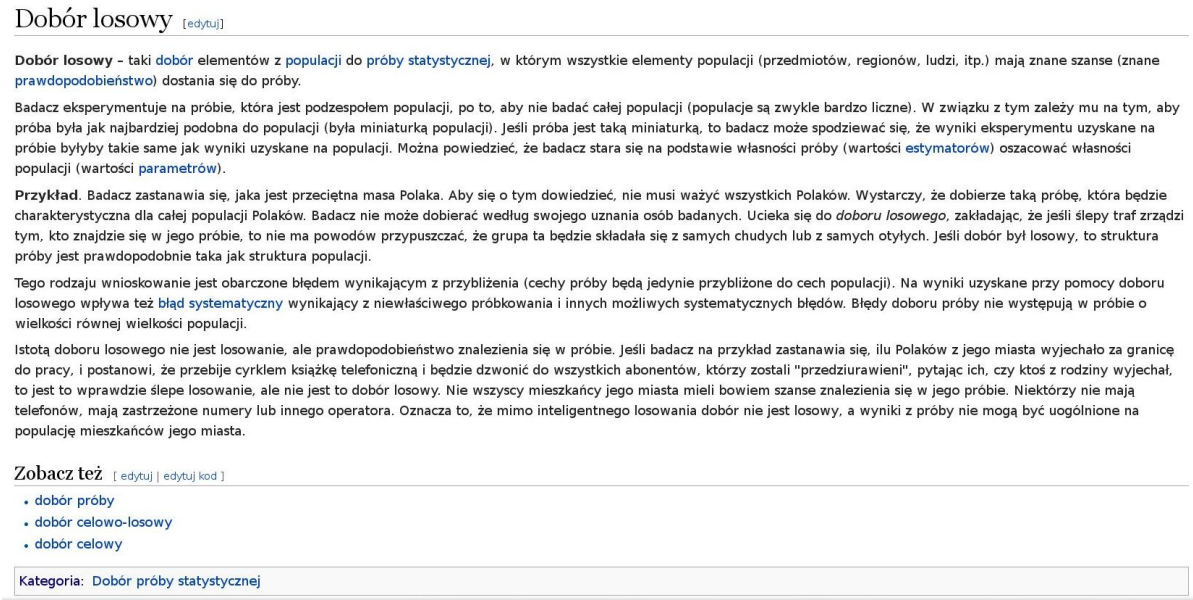
1. Wstępne przetwarzanie danych.
2. Utworzenie skupień „podobnych" słów.
3. Stworzenie macierzy o wymiarach liczba artykułów $\times$ liczba słów, gdzie wartością jest liczność występowania danego słowa w artykule.

[1] `www.wikipedia.org`

4. Użycie algorytmu k-średnich do podzielenia tekstów na skupienia.

3.1. Opis danych

Skuteczność algorytmu automatycznej kategoryzacji tematycznej testowana jest na artykułach polskiej wersji Wikipedii. Omówimy teraz rzeczony zbiór danych.



Dobór losowy [edytuj]

Dobór losowy - taki dobór elementów z populacji do próby statystycznej, w którym wszystkie elementy populacji (przedmiotów, regionów, ludzi, itp.) mają znane szanse (znane prawdopodobieństwo) dostania się do próby.

Badacz eksperymentuje na próbie, która jest podzespołem populacji, po to, aby nie badać całej populacji (populacje są zwykle bardzo liczne). W związku z tym zależy mu na tym, aby próba była jak najbardziej podobna do populacji (była miniaturką populacji). Jeśli próba jest taką miniaturką, to badacz może spodziewać się, że wyniki eksperymentu uzyskane na próbie byłyby takie same jak wyniki uzyskane na populacji. Można powiedzieć, że badacz stara się na podstawie własności próby (wartości estymatorów) oszacować własności populacji (wartości parametrów).

Przykład. Badacz zastanawia się, jaka jest przeciętna masa Polaka. Aby się o tym dowiedzieć, nie musi ważyć wszystkich Polaków. Wystarczy, że dobierze taką próbę, która będzie charakterystyczna dla całej populacji Polaków. Badacz nie może dobierać według swojego uznania osób badanych. Ucieka się do *doboru losowego*, zakładając, że jeśli ślepy traf zrządzi tym, kto znajdzie się w jego próbie, to nie ma powodów przypuszczać, że grupa ta będzie składała się z samych chudych lub z samych otyłych. Jeśli dobór był losowy, to struktura próby jest prawdopodobnie taka jak struktura populacji.

Tego rodzaju wnioskowanie jest obarczone błędem wynikającym z przybliżenia (cechy próby będą jedynie przybliżone do cech populacji). Na wyniki uzyskane przy pomocy doboru losowego wpływa też błąd systematyczny wynikający z niewłaściwego próbkowania i innych możliwych systematycznych błędów. Błędy doboru próby nie występują w próbie o wielkości równej wielkości populacji.

Istotą doboru losowego nie jest losowanie, ale prawdopodobieństwo znalezienia się w próbie. Jeśli badacz na przykład zastanawia się, ilu Polaków z jego miasta wyjechało za granicę do pracy, i postanowi, że przebije cyrklem książkę telefoniczną i będzie dzwonić do wszystkich abonentów, którzy zostali "przedziurawieni", pytając ich, czy ktoś z rodziny wyjechał, to jest to wprawdzie ślepe losowanie, ale nie jest to dobór losowy. Nie wszyscy mieszkańcy jego miasta mieli bowiem szanse znalezienia się w jego próbie. Niektórzy nie mają telefonów, mają zastrzeżone numery lub innego operatora. Oznacza to, że mimo inteligentnego losowania dobór nie jest losowy, a wyniki z próby nie mogą być uogólnione na populację mieszkańców jego miasta.

Zobacz też [edytuj | edytuj kod]

- dobór próby
- dobór celowo-losowy
- dobór celowy

Kategoria: Dobór próby statystycznej

Rysunek 3.1: Przykładowy artykuł z portalu Wikipedia. Na niebiesko wyróżnione zostały linki do innych tekstów z portalu. Źródło: `http://pl.wikipedia.org/wiki/Dob%C3%B3r_losowy`

Zgromadzone dane to zbiór 1 075 568 artykułów z polskiej Wikipedii z dnia 2. listopada 2014 roku w postaci plików XML [2]. Teksty składają się z treści sformułowanych w języku naturalnym, wzorów, kodów, linków wewnętrznych Wikipedii, linków do źródeł zewnętrznych, odniesień do źródeł bibliograficznych, cytatów, przypisów, rysunków (zdjęć, wykresów) wraz z podpisami, tabel, komentarzy, uwag, spisu treści, sekcji „Zobacz też", kategorii oraz znaczników typowych dla plików HTML-owych. Każdy z wyżej wymienionych elementów jest wyróżniony w tekście w inny sposób (np. linki wewnętrzne Wikipedii zawsze znajdowały się wewnątrz podwójnych nawiasów kwadratowych), co nie pozostaje bez znaczenia przy wstępnym przetwarzaniu danych.

3.2. Wstępne przetwarzanie danych

Ważnym elementem przy pracy z danymi jest ich wstępna obróbka, szczególnie gdy są to dane tekstowe. Jednocześnie nie ma ogólnych, odpowiednich dla wszystkich zagadnień, reguł postępowania – schemat działania trzeba dostosować pod konkretny problem i posiadane dane. Naturalnie kwestię wstępnej obróbki danych można pominąć, jednak wiąże się to z ryzykiem negatywnego wpływu na działanie algorytmu. Co więcej w przypadku gdy szczególnie istotne są słowa znajdujące się w tekście, przygotowanie danych może okazać się jedną z kluczowych kwestii, jeśli chodzi o jakość działania algorytmu.

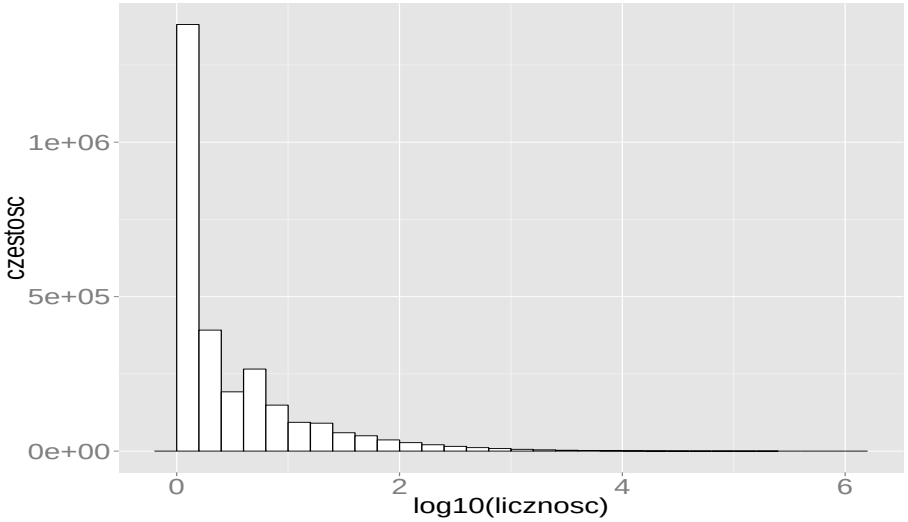
[2] Źródło: `http://dumps.wikimedia.org/plwiki/20141102/`

Jak wspomniano wcześniej pobrane dane składają się w dużej mierze z treści sformułowanych w języku naturalnym, jak i zawartości technicznej takiej jak linki czy znaczniki HTML-owe. Określenie istotności treści zawartej w poszczególnych częściach jest zasadniczym problemem przy wstępnej obróbce danych. Przyjmijmy, że część związaną z językiem naturalnym będziemy nazywać „tekstową", a pozostałe części tekstu – „techniczną". Związane są z nimi następujące aspekty (zaznaczmy, że rozważania te wymagają nieformalnego podejścia, intuicji oraz początkowego przejrzenia tekstów, co jest nieodłączną częścią praktycznej analizy danych):

- Część tekstowa zawiera główny opis artykuły, można się więc spodziewać, że w tej części zawarte zostanie meritum tekstu.
- Linki składają się ze słowa, które pojawia się w tekście, jak i odniesienia do innej strony. Ta pierwsza część może więc zawierać dużo informacji o temacie tekstu.
- Wzory oraz kody mogą dużo powiedzieć o tematyce artykułu (np. bardzo łatwo odróżnić wzór chemiczny od matematycznego), jednak w większości składają się one ze krótkich ciągów znaków (np. pojedynczych liter), które mogą być charakterystyczne dla wielu problemów.
- Cytaty mogą być ważne, choć często mogą mocno odbiegać od głównej tematyki tekstu.
- Przypisy są dodatkową informacją zawartą w tekście, często poruszające tematy poboczne.
- Odniesienia bibliograficzne, choć ważne z punktu widzenia wartości treści zawartych w artykule, nie wnoszą istotnych informacji o tematyce artykułu.
- Część artykułów zawiera bardzo dużo tabel, które czasem stanowią niemal jedyną treść. Jednakowoż służą one uporządkowaniu wiedzy i treść w nich zawarta często nie wnosi istotnych informacji o tematyce tekstu, zawierając jedynie słowa hasłowe bądź wylistowania danych zagadnień.
- Podpisy pod rysunkami są zazwyczaj powtórzeniem zdań bądź ich fragmentów z części opisowej.
- Spis treści zawiera tytuły, które są następnie powtórzone w tekście.
- Sekcja „Zobacz też" może być ważna, gdyż wskazuje na połączenia tematyczne między tekstami.
- Podobnie kategorie wskazują w szczególności na tematy poruszanego zagadnienia.
- Znaczniki HTML-owe oraz wyróżnienia powyższych elementów nie wnoszą żadnej informacji o tematyce tekstu i służą jedynie odpowiedniemu wyświetleniu treści.

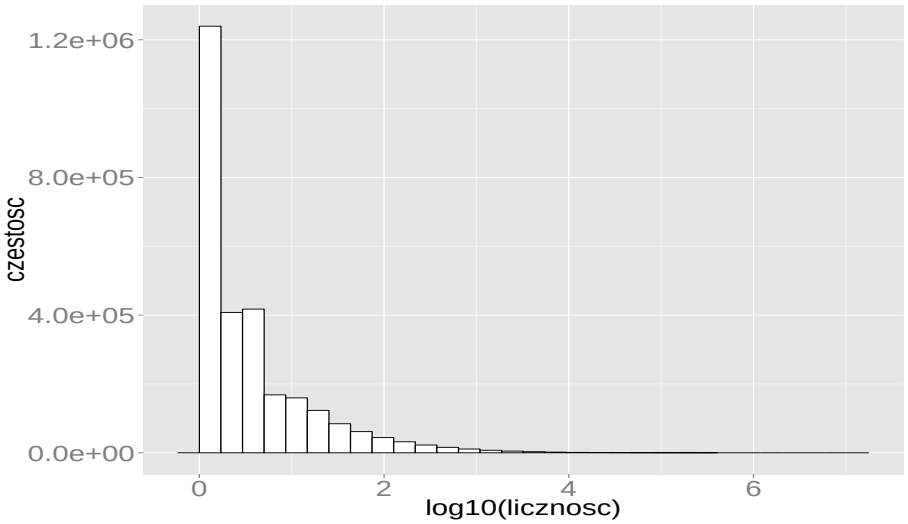
Wstępną obróbkę danych przeprowadzamy uwzględniając powyższe aspekty. Początkowo z części tekstowej usuwamy wszystkie znaki nie będące literami alfabetu łacińskiego. Poddajemy jej dalszej obróbce po przetworzeniu części technicznej. Słowa, które występują w tekście, a pod którymi znajduje się link, pozostawiamy bez zmian. Wzory oraz kody usuwamy w całości ze względu na trudność w rozróżnieniu czego podany fragment dotyczy oraz ze względu na krótkie znaki, które zawierają. Cytaty pozostawiamy bez zmian. Przypisy usuwamy z artykułów, jako że zawierają jedynie dodatkową z punktu widzenia głównego tekstu, treść. Źródła bibliograficzne usuwamy w całości. Teksty oczyszczamy także z tabel, rysunków, podpisów pod nimi oraz spisu treści. Sekcję „Zobacz też" oraz kategorie pozostawiamy jako potencjalne

źródło podobnej tematyki do tej zawartej w tekście. Wszelkie znaczniki HTML-owe usuwamy jako bezwartościowe z punktu widzenia treści artykułu. Podobnie teksty oczyszczamy z tytułów, które pojawiają się w większości tekstów, tj. *Zobacz też, Linki zewnętrzne* oraz *Bibliografia*.



Rysunek 3.2: Histogram logarytmu dziesiętnego z liczby artykułów, w których dane słowo występuje.

Tak otrzymane teksty dzielimy na słowa, które przekształcamy do wyrazów o małych literach. Nie chcemy rozróżniać słów ze względu na wielkość liter, gdyż nie powinna ona mieć znaczenia dla tematyki treści. Do każdego tekstu dodajemy informację o tym, jakie słowa i w jakich licznościach w nim występują. W ten sposób otrzymujemy 2 806 765 różnych słów we wszystkich, tj. w 1 075 568, artykułach. 49% słów występuje tylko w jednym tekście (p. rys. 3.2). Nieco ponad 3.5% słów znajduje się w stu i więcej artykułach, natomiast jedynie 0.6% wszystkich wyrazów pojawia się w więcej niż tysiącu tekstów.



Rysunek 3.3: Histogram logarytmu dziesiętnego z liczby wystąpień słów we wszystkich artykułach.

44% słów występuje dokładnie raz we wszystkich artykułach (p. rys. 3.3). Prawie 4.3% wyrazów pojawia się ponad sto razy, natomiast mniej niż jeden procent słów występuje tysiąc i więcej razy.

Słowa występujące tylko w jednym tekście to zazwyczaj słowa w obcych językach, przykładowo: `jużdortransstroj, youtsos, odety, knežlaz, pallebitzke, rulicach, werkowie, rumilla, metyklotiazyd, bazelak`, choć czasem są to słowa będące odmianą słów bardziej częściej występujących, np. słowo `uchybiają` jest odmianą czasownika `uchybiać`. Takie słowa warto wziąć pod uwagę przy analizie, gdyż są „podobne" do słów częściej występujących, a więc mogą polepszyć jakość dopasowania pod względem tematycznym.

	Słowo	Liczba artykułów	Liczba wystąpień
1	w	1 003 961	10 330 250
2	i	726 209	4 290 653
3	na	689 559	3 215 428
4	z	649 564	3 252 352
5	do	559 672	2 297 910
6	się	537 360	2 094 912
7	roku	420 178	1 292 537
8	a	378 941	919 278
9	od	378 327	935 799
10	jest	343 846	993 143
11	przez	336 596	852 463
12	oraz	288 718	663 760
13	po	286 818	765 075
14	o	273 574	674 431
15	ur	241 888	365 934
16	to	234 629	527 121
17	jako	232 919	663 364
18	latach	232 017	380 362
19	był	229 219	503 938
20	został	228 985	509 705

Tabela 3.1: Lista dwudziestu najczęściej występujących słów. Kolumna oznaczona jako *Liczba artykułów* oznacza liczbę tekstów, w których wystąpiło dane słowo, natomiast ostatnia kolumna mówi o ilości wystąpień wyrazu ogółem we wszystkich artykułach.

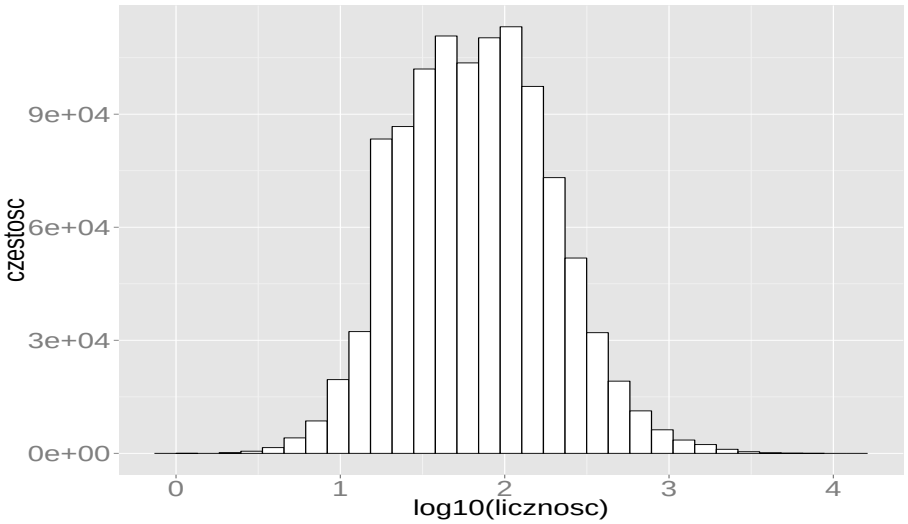
Słowa najczęściej występujące prezentuje tabela 3.1. W większości przypadków są to słowa nie istotne w kontekście analizy tematycznej tekstu (ang. *stopwords*). Takich słów można wyróżnić więcej, np.

`ach, aj, albo, bardzo, bez, bo, być, ci, cię, ciebie, co, czy, daleko, dla, dlaczego, dlatego, do, dobrze, dokąd, dość, dużo, dwa, dwaj, dwie, dwoje, dziś, dzisiaj, gdyby, gdzie, go, ich, ile, im, inny, ja, ją, jak, jakby, jaki, je, jeden, jedna, jedno, jego, jej, jemu, jeśli, jest, jestem, jeżeli, już, każdy, kiedy, kierunku, kto, ku, lub, ma, mają, mam, mi, mną, mnie, moi, mój, moja, moje, może, mu, my, na, nam, nami, nas, nasi, nasz, nasza, nasze, natychmiast, nią, nic, nich, nie, niego, niej, niemu, nigdy, nim, nimi, niż, obok, od, około, on, ona, one, oni, ono, owszem, po, pod, ponieważ, przed, przedtem, są, sam, sama, się, skąd, tak, taki, tam, ten, to, tobą, tobie, tu, tutaj, twoi, twój, twoja, twoje, ty, wam, wami, was, wasi, wasz,`

`wasza, wasze, we, więc, wszystko, wtedy, wy, żaden, zawsze, że`

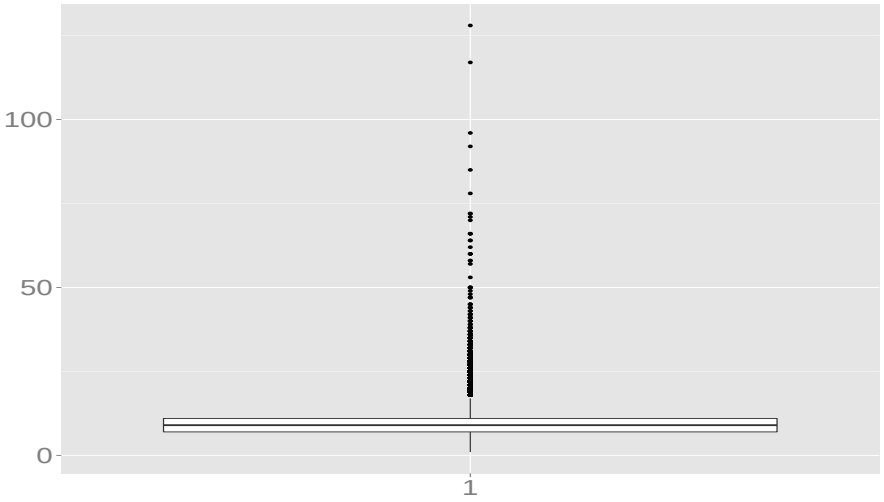
Te i podobne słowa oraz wyrazy jedno- i dwuznakowe, należy usunąć ze zbioru danych, gdyż nie wnoszą żadnej istotnej informacji o tematyce tekstu. Dokonujemy tego w kolejnym etapie wstępnej obróbki tekstów. Łącznie ze zbioru słów usuwamy 1 822 wyrazy.

Średnia liczba unikalnych słów występujących w artykule wynosi 121, mediana to zaledwie 66 unikalnych wyrazów. W 2 272 tekstach wystąpiło mniej niż pięć słów. Artykuły te zawierały przede wszystkim tabele i rysunki, stąd po wstępnej obróbce danych pozostało w nich niewiele wyrazów. Histogram logarytmu dziesiętnego liczby unikalnych słów w tekstach przedstawia rys. 3.4.



Rysunek 3.4: Histogram logarytmu dziesiętnego liczby unikalnych słów w artykule.

Mediana długości słów wynosi 9, średnia jest nieco wyższa (p. rys. 3.5). Najkrótsze występujące słowa są jednoznakowe, jest ich 249. Najdłuższe słowo ma 128 znaków. Słowa o długości ponad 11 znaków stanowią 19.5% wszystkich słów.



Rysunek 3.5: TO DO : ZMIEN TEN WYKRES NA HISOGRAM. Wykres skrzynkowy długości słów.

3.3. Utworzenie skupień „podobnych” słów / Jakiś mądry tytuł

Po wstępnej obróbce danych dostajemy 2 806 765 unikalnych słów, z czego blisko połowa występuje jedynie raz we wszystkich tekstach. Algorytm dzielący teksty tematycznie bierze pod uwagę liczności słów, które znajdują się w artykule a daną wejściową jest macierz o wymiarach liczba artykułów × liczba słów. Jeśliby nie przetworzyć dalej danych, to macierz ta miałaby wymiary 1 075 568 × 2 806 765, gdzie przeważająca większość rekordów byłaby równa zeru. Algorytm mógłby nie poradzić sobie z tak dużą ilością danych, zwłaszcza gdy większa część rekordów jest „niewypełniona”. Stąd zachodzi potrzeba zmniejszenia wymiaru danych.

Ponieważ rekordy składają się ze słów, można wyznaczyć skupienia wyrazów „podobnych” do siebie. Podobieństwo słów można określić za pomocą odległości na przestrzeni ciągów znaków opisanych w rozdziale 1. Następnie dany tekst można przedstawić jako sumę liczby wystąpień słów z danego skupienia, zamiast liczby wystąpień pojedynczych słów. Przykładowo, jeśli skupienie A składa się ze słów s, t, u, które wystąpiły w danym artykule, odpowiednio, x, y, z razy, to słowa ze skupienia A wystąpiły w tym tekście łącznie $x + y + z$ razy. Dzieląc wszystkie wyrazy na skupienia, możemy znacząco zmniejszyć liczbę wyrazów, biorąc pod uwagę liczności grup słów, zamiast liczności pojedynczych wyrazów.

Aby podzielić zbiór słów na skupienia, najbardziej naturalne wydaje się zbudowanie macierzy odległości wszystkich słów od siebie i zastosowanie algorytmu aglomeracyjnego. Jednakowoż wiąże się to z dużą złożonością pamięciową i obliczeniową, stąd też podejście takie nie zostało wykorzystane w niniejszej pracy. Inny pomysł polega na przyłączaniu do skupienia słów dopóty, dopóki średnia odległość w zbiorze nie przekroczy zadanej liczby. Wstępne testy na losowej próbce tysiąca słów wykazały, że jakość takiego podziału jest słaba, a czas obliczeń względnie długi.

Stemming. Stąd postanowiono dokonać podziału zbioru słów w inny sposób. Początkowo przeprowadzamy tzw. *stemming*, czyli sprowadzenie słowa do jego rdzenia. Odmiana słowa nie zmienia jego tematyki, a dzięki takiemu podejściu możemy znacznie ograniczyć liczbę unikalnych słów w zbiorze. Przykładowo słowa `zjednoczonych`, `zjednoczyli`, `drużynom`, `drużynie` zostaną sprowadzone odpowiednio do form `zjednoczyć`, `drużyna`. Dzięki takiemu podejściu, każde skupienie będzie miało swoje *słowo-reprezentanta*, które jednoznacznie charakteryzuje podzbiór. Będziemy je nazywać słowem-reprezentantem lub po prostu *reprezentantem*. W ten sposób do odpowiedniego skupienia możemy odnosić się poprzez jego reprezentanta.

	Język	Liczba słów	Procent ogółu
1	polski	664 315	23.7
2	angielski	41 087	1.5
3	niemiecki	21 117	0.8
4	francuski	20 438	0.7
5	ogółem	746 942	26.6

Tabela 3.2: Liczba słów na których zastosowano *stemming* w poszczególnych językach

.

Do przeprowadzenia *stemmingu* używamy programu *Hunspell* [3] – korektora pisowni i analizatora morfologicznego używany w wielu programach typu *open source*. W ten sposób grupujemy 746 942 słów, co stanowi ok. 27% wszystkich słów, w 186 958 skupienia. Ponieważ część wyrazów stanowią słowa obcojęzyczne przeprowadzamy *stemming* w języku polskim, angielskim, niemieckim oraz francuskim (p. tabela 3.2). Przykładowe skupienia prezentuje tabela 3.3. Warto zauważyć, że słowa w podzbiorach są podobne tematycznie, choć do skupienia o reprezentancie `główny` trafiły wyrazy o znaczeniu przeciwnym. Widać więc, że podział taki nie jest idealny.

Reprezentant	Słowa w skupieniu
czas	czasie, czasach, czas, czasom
główny	głównie, główne, główną, głównych, głównego, głównym, główna, głównymi, główny, głównej, główni, głównemu, niegłówny, niegłównym, niegłówne, niegłównych, niegłówną
miał	miał, miały, miałem, miału, miałach, miale, miałów, miałami, miałom
nazwa	nazwa, nazwę, nazwy, nazwą, nazwie, nazw, nazwami, nazwach, nazwom
nr	nr, nry, nru, nrem, nrze, nrów
osoba	osób, osoby, osoba, osobą, osobę, osobom, osobie, osobami, osobach, osobo
udział	udział, udziału, udziałem, udziale, udziały, udziałów, udziałami, udziałach, udziałom
wieś	wsi, wsie, wieś, wsią, wsiach, wsiami, wsiom
zostać	został, została, zostały, zostało, zostanie, zostali, zostać, zostaną, zostania, zostałby, zostałaby, zostałyby, zostaniu, zostałem, zostałoby, zostaniesz, zostanę, zostaliby, zostaliśmy, zostaniemy, zostaniem, zostałam, zostałeś, zostaliście, zostaniecie, zostałaś, zostałbym, zostalibyśmy, zostałbyś, zostano, zostałabym

Tabela 3.3: Przykładowe skupienia uzyskane przy pomocy *stemmingu*.

Podział przy użyciu metryk. Tak zaproponowany podział słów na skupienia wykorzystuje jedynie ok. 27% zbioru wszystkich wyrazów. Co więcej większość z podzbiorów jest zaledwie kilkuelementowa (p. tabela 3.4).

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	1	2	4	5	89

Tabela 3.4: Rozkład liczby słów w skupieniu.

Stąd można zastosować następujące schematy postępowania, które po pierwsze zredukują liczbę używanych grup słów, a po drugie wykorzystają dodatkowo zbiór niepogrupowanych wyrazów. Procedury te polegają na [PLIK 14]:

1. Dołączeniu do skupień słów jeszcze niepogrupowanych.
2. Dołączeniu do skupień zawierających pięć i więcej elementów, podzbiorów o mniejszej liczności.
3. Zastosowaniu najpierw punktu 1, a następnie punktu 2.

Omówimy teraz bliżej na czym polegają powyższe kroki.

[3] `http://hunspell.sourceforge.net/`

Procedura 1. W kroku tym chcemy użyć większej liczby dostępnych słów. Dzięki temu zwiększą się liczności występowania grup słów w tekście, co być może polepszy jakość podziału artykułów na skupienia. Algorytm ten jednak nie zmieni liczby skupień.

Schemat działania jest następujący: bierzemy słowo nieprzydzielone do żadnego skupienia. Liczymy odległość tego wyrazu (przy użyciu odległości zdefiniowanych w rozdziale 1) od wszystkich słów-reprezentantów dotychczasowo otrzymanych skupień. Słowo przydzielamy do tego podzbioru, do którego reprezentanta było mu najbliżej (p. algorytm 4). Jeśli wyraz ma taką samą (najmniejszą) odległość do kilku reprezentantów, wybieramy pierwszy w kolejności. Jeśli odległość słowa do wszystkich reprezentantów jest nieokreślona lub równa nieskończoności, to wyraz ten pomijamy, tj. nie dodajemy go do żadnego ze skupień. Taka sytuacja może się zdarzyć, w przypadku zastosowania odległości opartych na q-gramach, gdy długość słowa jest mniejsza od q.

Algorithm 4 Algorytm przydzielający niepogrupowane słowo do skupienia.

given: data set of representants R, data set of words to categorize W, metric d
for $w \in W$ **do**
    $C(w) = \arg \min_{i:\ r_i \in R} d(w, r_i)$
    if $\text{length}(C(w)) > 1$ **then**
        $C(w) = C(w)[1]$
    end if
end for

Procedura 2. W tym kroku chcemy zredukować liczbę uzyskanych skupień. Dzięki temu zwiększą się liczności występowania grup słów w tekście, a ponadto zmniejszy się liczba skupień, co może przyczynić się do lepszego działania algorytmu dzielącego teksty na skupienia.

Algorithm 5 Algorytm łączący małe i duże skupienia.

given: data set of representants R, vector of clusters' size $\mathbf{s}$, metric d
$R_m = \emptyset, R_d = \emptyset$
for $r \in R$ **do**
    if $s_r \leq 5$ **then**
        $R_m = R_m \cup r$
    else
        $R_d = R_d \cup r$
    end if
end for
for $w \in C_m$ **do**
    if $|w| < 4$ **then**
        continue / next
    end if
    $C(w) = \arg \min_{i:\ r_i \in R_d} d(w, r_i)$
    if $\text{length}(C(w)) > 1$ **then**
        $C(w) = C(w)[1]$
    end if
end for

Schemat działania jest następujący (p. algorytm 5): sprawdzamy, jakie są liczności wszystkich skupień. Jeśli liczność skupienia jest większa od pięciu, to taki podzbiór oznaczamy jako

„duży". Do takich skupień będziemy przyłączać mniejsze podgrupy. Jeśli liczność skupienia jest mniejsza lub równa 5, to podzbiór oznaczamy jako „mały". Takie skupienie będziemy przyłączać do podzbiorów „dużych". Te pierwsze skupienia nazwijmy dużymi skupieniami, natomiast te drugie – małymi.

Mając tak podzielone skupienia, weźmy reprezentantów małych podzbiorów. Jeśli długość słowa-reprezentanta nie przekracza trzech znaków, to skupienie takie pomijamy w dalszej analizie. Ma to na celu uniknięcie analizy słów, które nie mają znaczenia, jak zbitek dwóch lub trzech takich samych liter (np. `aa` lub `bbb`). Następnie postępowanie jest podobne jak w algorytmie 4: liczymy odległość reprezentanta małego skupienia od wszystkich słów-reprezentantów dużych skupień. Sprawdzamy, która z wyliczonych odległości była najmniejsza. Podzbiór, którego reprezentantem jest analizowane słowo, przydzielamy do tego skupienia, do którego reprezentanta było mu (słowu) najbliżej. Jeśli wyraz ma taką samą (najmniejszą) odległość do kilku reprezentantów, wybieramy pierwszego w kolejności.

Procedura 3. Krok trzeci polega na wykonaniu najpierw procedury pierwszej, a następnie drugiej.

Wybór odległości. Mając trzy powyższe algorytmy, możemy przystąpić do dalszej obróbki zbioru słów. Zanim to jednak nastąpi należy wybrać odległości, dzięki którym będzie to możliwe. W rozdziale 1 przedstawiono pięć odległości opartych na operacjach edycyjnych, trzy odległości oparte na q-gramach oraz dwie miary heurystyczne.

Odległość Hamminga odrzucamy, gdyż można ją zastosować jedynie na napisach o tej samej długości. Odległości najdłuższego wspólnego podnapisu, Levenshteina, optymalnego dopasowania napisów i Damerau-Levenshteina różnią się jedynie zbiorem bazowych operacji edycyjnych, często dając tę samą odległość. Stąd postanowiliśmy użyć dwóch „skrajnych" odległości, tj. takich, które pozwalają na najmniejszą i największą liczbę bazowych operacji edycyjnych, czyli odległość najdłuższego wspólnego podnapisu i Damerau-Levenshteina.

Z odległości opartych na q-gramach wyselekcjonowaliśmy odległość Jaccarda oraz q-gramową jako najbardziej reprezentatywne. W obu przypadkach wybraliśmy $q = 4$. Dzięki takiemu podejściu unikniemy przetwarzania słów o długości mniejszej niż cztery znaki.

Miary heurystyczne pominęliśmy.

Otrzymane zbiory. [GDZIES TU NAPISAC, ZE UZYWALAM R-A?] Na zbiorze skupień otrzymanym po wykonaniu *stemmingu* zastosowano trzy powyższe algorytmy przy użyciu każdej z czterech odległości, dostając łącznie 13 różnych zbiorów skupień (wliczając w to zbiór, otrzymany ze *stemmingu*). Zbiory te oznaczmy jako *clust_X* gdzie X jest przyrostkiem oznaczającym algorytm i zastosowaną odległość. Metodologia nazewnictwa jest następująca: w przypadku, gdy dołączaliśmy słowa do istniejących skupień (tj. zastosowany był algorytm 4 / procedura 1) dodajemy jedynie przyrostek oznaczający zastosowaną odległość, tj. *lcs, dl, jac* lub *qg*, np. *clust_lcs*. Jeśli użyliśmy algorytmu 5 / procedury 2, zmniejszającego liczbę skupień, to dodajemy przyrostek *red_* oraz zastosowaną odległość, np. *clust_red_lcs*. Jeśli oba algorytmy zostały zastosowane, to łączymy je w nazwie, dostając np. *clust_lcs_red_lcs*. Zbiór otrzymany po wykonaniu *stemmingu* oznaczamy po prostu *clust*.

Liczbę skupień oraz liczbę słów zawartą w skupieniu dla poszczególnych zbiorów zawiera tabela 3.5. Zbiory, na których zastosowano algorytm 4 lub jedynie *stemming* zawierają 186 958 skupień, co dało redukcję ok. 93% względem oryginalnego zbioru słów (tj. 2 806 765). W skupieniach tych znajduje się od prawie 750 000 do ponad 1 000 000 słów, czyli między 27% a 38% wszystkich wejściowych wyrazów. Druga grupa zbiorów, tj. taka, która jest wynikiem działania algorytmu 5 zawiera dokładnie 43 919 skupienia, co daje redukcję równą 98.4%.

Słowa zawarte w tych skupieniach stanowią ok. 26% wejściowego zbioru wyrazów. Trzecia grupa zbiorów, ma nieco mniejszą redukcję niż poprzednia i wynosi prawie 98%, zawierając jednocześnie ok. 38% wszystkich wyrazów.

	Zbiór	Liczba skupień	Redukcja	Liczba słów	Procent
1	clust	186 958	93.3%	746 957	27%
2	clust_lcs	186 958	93.3%	1 080 260	38%
3	clust_dl	186 958	93.3%	1 080 260	38%
4	clust_jaccard	186 958	93.3%	1 070 750	38%
5	clust_qgram	186 958	93.3%	1 070 750	38%
6	clust_red_lcs	43 919	98.4%	743 053	26%
7	clust_red_dl	43 919	98.4%	743 053	26%
8	clust_red_jaccard	43 919	98.4%	739 338	26%
9	clust_red_qgram	43 919	98.4%	739 338	26%
10	clust_lcs_red_lcs	65 350	97.7%	1 037 393	37%
11	clust_dl_red_dl	66 378	97.6%	1 060 474	38%
12	clust_jaccard_red_jaccard	69 570	97.5%	1 063 131	38%
13	clust_qgram_red_qgram	62 434	97.8%	1 063 131	38%

Tabela 3.5: Zbiory skupień wraz z ich liczbą oraz liczbą słów w skupieniu. Redukcja oznacza procent zredukowania z wejściowego zbioru słów do liczby otrzymanych skupień. Ostatnia kolumna mówi ile procent wszystkich słów zbioru wejściowego znajduje się w skupieniu.

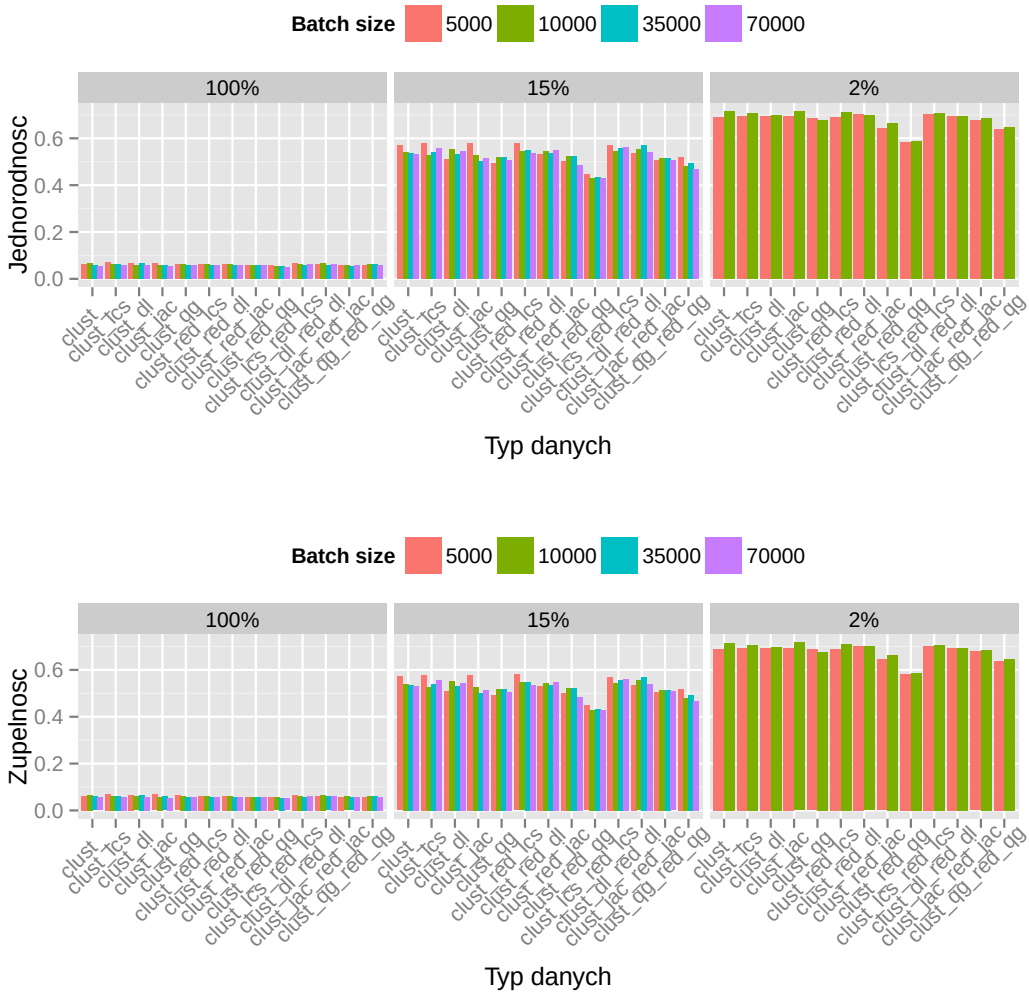
3.4. Podział tekstów

Mając tak zdefiniowane skupienia słów możemy przystąpić do podziału zbioru artykułów. Do tego celu użyjemy algorytmu mini-wsadowego k-średnich (algorytm 3 z rozdziału 2). Przypomnijmy, że algorytm ten jest metodą pośrednią pomiędzy algorytmem wsadowym, który w każdej iteracji opiera się na wszystkich obserwacjach, a algorytmem SGD, biorącym w każdej iteracji po jednej obserwacji ze zbioru.

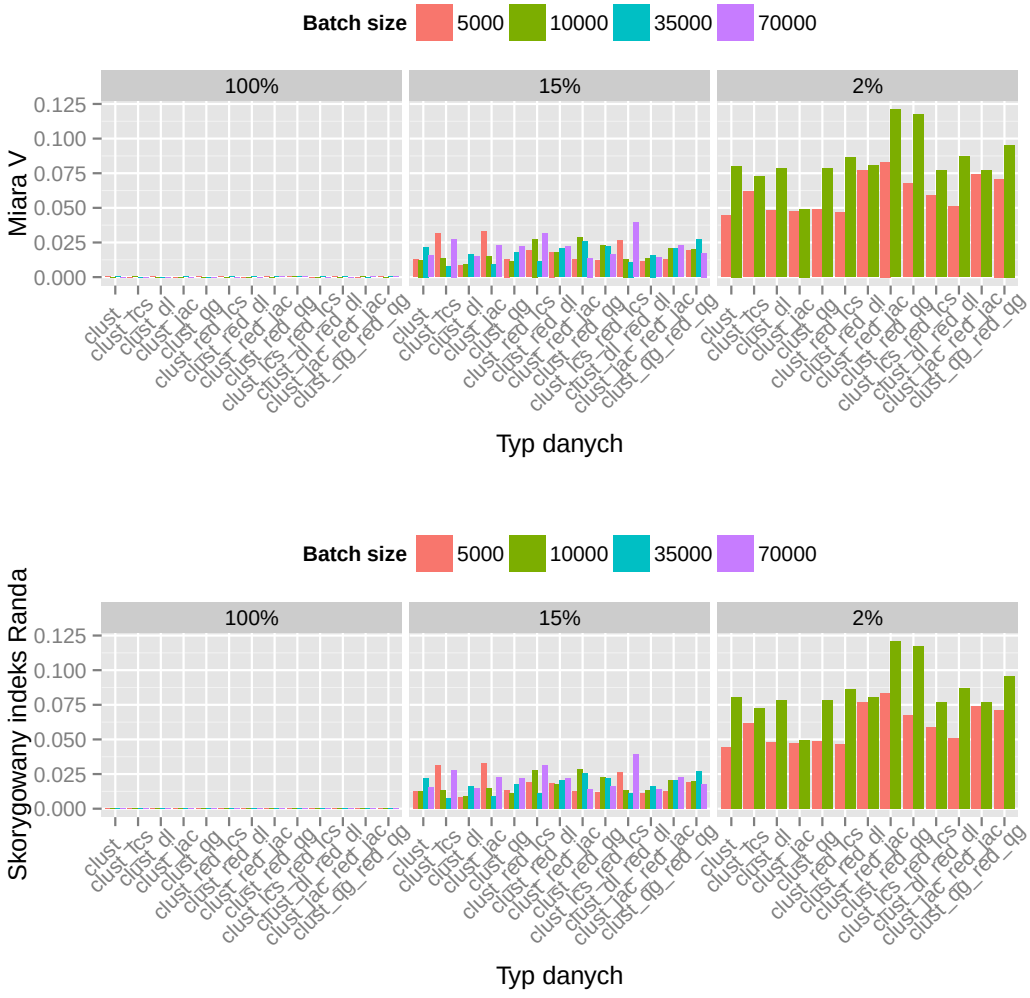
Aby więc użyć algorytmu mini-wsadowego musimy wybrać najpierw liczbę skupień k oraz parametr b, określający ile obserwacji będzie miało swój wkład w każdej iteracji. Zajmijmy się najpierw tą drugą wartością. Ponieważ nie wiemy jak bardzo jakość podziału zależy od parametru b, postanowiliśmy sprawdzić działanie algorytmu dla czterech wartości b: 5 000, 10 000, 35 000 oraz 70 000. Dostaniemy w ten sposób 52 wyniki analizy, oparte na 13 różnych zbiorach wejściowych.

Zanim określimy wartość parametru k, zastanówmy się w jaki sposób będziemy mierzyć jakość otrzymanych podziałów. Cztery na pięć zaprezentowanych miar w rozdziale 2 wymaga znajomości prawdziwego podziału zbioru. Przypomnijmy, że nasz zbiór danych to artykuły z polskiej Wikipedii, które mają określoną kategorię tematyczną. Można więc wykorzystać znaną nam wiedzę o kategoriach i na jej podstawie określić jakość podziału otrzymanego w wyniku działania algorytmu. Liczba różnych kategorii, które określają tematykę artykułów wynosi 56 283. Próba wykonania analizy skupień przy użyciu algorytmu mini-wsadowego z tak dużym k, zakończyła się niepowodzeniem, mimo posiadania dużej ilości pamięci RAM (ponad 100 GB). Stąd też nastąpiła potrzeba zredukowania tej liczby. Ponieważ struktura kategorii Wikipedii jest drzewiasta (por. `http://pl.wikipedia.org/wiki/Wikipedia:Drzewo_kategorii`), można zastąpić kategorię przypisaną do artykułu kategorią ogólniejszą. Po takiej redukcji otrzymano 6 922 grup tematycznych. Jednak rozkład liczby artykułów w

otrzymanych kategoriach był mocno skośny. Taka sytuacja jest silnie niesprzyjająca, gdyż chcemy mieć podobne liczności w grupach. Ręcznie podzielono zatem najbardziej liczne tematy na podtematy, natomiast te o najmniejszej liczności połączono zachowując przy tym podobieństwo tematyki. W ten sposób uzyskano sto różnych tematów o podobnym rozkładzie liczby artykułów. Wstępne testy wykazały, że tak otrzymane k pozwala na dokonanie obliczeń w relatywnie krótkim czasie i nie zajmując dużej ilości pamięci RAM.



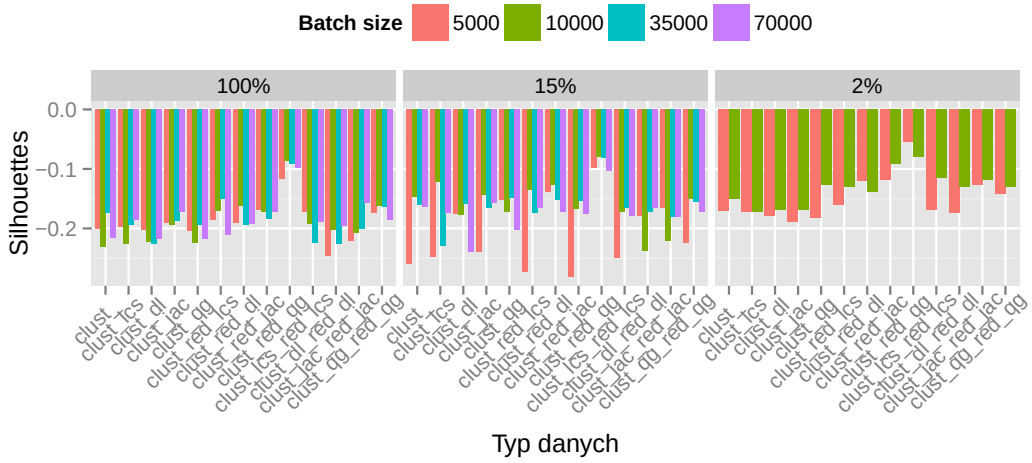
Rysunek 3.6: Jednorodność.



Rysunek 3.7: Miara V.

	Typ danych	Batch	Jedn.	Zup.	Miara V	ARI	Silhouettes
1	_	5000	0.19	0.51	0.27	0.01	-0.19
2	_	10000	0.19	0.50	0.28	0.01	-0.19
3	_	35000	0.22	0.53	0.31	0.02	-0.20
4	_	70000	0.22	0.52	0.31	0.02	-0.19
5	_lcs	5000	0.17	0.51	0.26	0.01	-0.22
6	_lcs	10000	0.17	0.51	0.26	0.01	-0.21
7	_lcs	35000	0.18	0.53	0.27	0.01	-0.21
8	_lcs	70000	0.19	0.52	0.28	0.01	-0.24
9	_dl	5000	0.20	0.51	0.28	0.01	-0.20
10	_dl	10000	0.19	0.51	0.27	0.01	-0.20
11	_dl	35000	0.21	0.52	0.30	0.02	-0.20
12	_dl	70000	0.22	0.53	0.31	0.01	-0.22
13	_jaccard	5000	0.18	0.53	0.27	0.01	-0.24
14	_jaccard	10000	0.19	0.50	0.27	0.01	-0.21
15	_jaccard	35000	0.19	0.51	0.28	0.01	-0.21
16	_jaccard	70000	0.21	0.51	0.30	0.01	-0.22
17	_qgram	5000	0.18	0.50	0.26	0.01	-0.22
18	_qgram	10000	0.18	0.47	0.26	0.02	-0.21
19	_qgram	35000	0.21	0.52	0.30	0.02	-0.22
20	_qgram	70000	0.19	0.49	0.28	0.01	-0.19
21	_red_lcs	5000	0.19	0.54	0.29	0.01	-0.21
22	_red_lcs	10000	0.20	0.51	0.28	0.01	-0.21
23	_red_lcs	35000	0.22	0.53	0.31	0.02	-0.13
24	_red_lcs	70000	0.24	0.52	0.32	0.03	-0.18
25	_red_dl	5000	0.20	0.53	0.29	0.01	-0.21
26	_red_dl	10000	0.21	0.51	0.29	0.02	-0.19
27	_red_dl	35000	0.21	0.51	0.30	0.02	-0.15
28	_red_dl	70000	0.24	0.53	0.33	0.02	-0.19
29	_red_jaccard	5000	0.20	0.48	0.28	0.01	-0.17
30	_red_jaccard	10000	0.22	0.51	0.31	0.02	-0.21
31	_red_jaccard	35000	0.22	0.51	0.30	0.02	-0.20
32	_red_jaccard	70000	0.22	0.51	0.31	0.01	-0.21
33	_red_qgram	5000	0.18	0.41	0.25	0.02	-0.10
34	_red_qgram	10000	0.19	0.42	0.26	0.02	-0.11
35	_red_qgram	35000	0.20	0.42	0.27	0.02	-0.12
36	_red_qgram	70000	0.20	0.42	0.27	0.02	-0.11
37	_red_lcs_lcs	5000	0.18	0.52	0.26	0.01	-0.23
38	_red_lcs_lcs	10000	0.18	0.57	0.27	0.01	-0.25
39	_red_lcs_lcs	35000	0.18	0.55	0.27	0.01	-0.23
40	_red_lcs_lcs	70000	0.23	0.57	0.33	0.01	-0.24
41	_red_dl_dl	5000	0.18	0.54	0.27	0.01	-0.18
42	_red_dl_dl	10000	0.21	0.57	0.31	0.01	-0.26
43	_red_dl_dl	35000	0.20	0.58	0.30	0.01	-0.26
44	_red_dl_dl	70000	0.23	0.57	0.33	0.01	-0.21
45	_red_jaccard_jaccard	5000	0.19	0.48	0.27	0.01	-0.15
46	_red_jaccard_jaccard	10000	0.20	0.50	0.29	0.01	-0.24
47	_red_jaccard_jaccard	35000	0.22	0.51	0.31	0.02	-0.21
48	_red_jaccard_jaccard	70000	0.23	0.52	0.32	0.02	-0.21
49	_red_qgram_qgram	5000	0.17	0.45	0.25	0.01	-0.21
50	_red_qgram_qgram	10000	0.19	0.46	0.26	0.01	-0.21
51	_red_qgram_qgram	35000	0.20	0.47	0.28	0.01	-0.20
52	_red_qgram_qgram	70000	0.18	0.45	0.26	0.01	-0.16

	Typ danych	Batch	Jedn.	Zup.	Miara V	ARI	Silhouettes
1	_	5000	0.02	0.06	0.03	0.00	-0.20
2	_	10000	0.03	0.06	0.04	0.00	-0.23
3	_	35000	0.03	0.06	0.04	0.00	-0.17
4	_	70000	0.02	0.06	0.03	0.00	-0.22
5	_lcs	5000	0.02	0.07	0.04	0.00	-0.20
6	_lcs	10000	0.02	0.06	0.04	0.00	-0.23
7	_lcs	35000	0.03	0.06	0.04	0.00	-0.19
8	_lcs	70000	0.02	0.06	0.03	-0.00	-0.19
9	_dl	5000	0.02	0.06	0.04	0.00	-0.20
10	_dl	10000	0.02	0.06	0.03	0.00	-0.22
11	_dl	35000	0.02	0.07	0.03	-0.00	-0.23
12	_dl	70000	0.03	0.06	0.04	0.00	-0.22
13	_jaccard	5000	0.02	0.07	0.03	0.00	-0.19
14	_jaccard	10000	0.02	0.06	0.03	0.00	-0.19
15	_jaccard	35000	0.03	0.06	0.04	0.00	-0.19
16	_jaccard	70000	0.02	0.05	0.03	0.00	-0.17
17	_qgram	5000	0.02	0.06	0.04	0.00	-0.20
18	_qgram	10000	0.02	0.06	0.03	0.00	-0.22
19	_qgram	35000	0.02	0.06	0.03	0.00	-0.19
20	_qgram	70000	0.03	0.06	0.04	0.00	-0.22
21	_red_lcs	5000	0.03	0.06	0.04	0.00	-0.19
22	_red_lcs	10000	0.03	0.06	0.04	0.00	-0.17
23	_red_lcs	35000	0.03	0.06	0.04	0.00	-0.15
24	_red_lcs	70000	0.02	0.06	0.03	0.00	-0.21
25	_red_dl	5000	0.02	0.06	0.03	0.00	-0.19
26	_red_dl	10000	0.03	0.06	0.04	0.00	-0.16
27	_red_dl	35000	0.03	0.06	0.04	0.00	-0.19
28	_red_dl	70000	0.02	0.06	0.03	0.00	-0.19
29	_red_jaccard	5000	0.03	0.06	0.04	0.00	-0.17
30	_red_jaccard	10000	0.03	0.06	0.04	0.00	-0.17
31	_red_jaccard	35000	0.03	0.06	0.04	0.00	-0.18
32	_red_jaccard	70000	0.03	0.06	0.04	0.00	-0.17
33	_red_qgram	5000	0.02	0.06	0.03	0.00	-0.12
34	_red_qgram	10000	0.02	0.06	0.03	0.00	-0.09
35	_red_qgram	35000	0.03	0.05	0.03	0.00	-0.09
36	_red_qgram	70000	0.02	0.05	0.03	0.00	-0.10
37	_red_lcs_lcs	5000	0.03	0.06	0.04	0.00	-0.17
38	_red_lcs_lcs	10000	0.02	0.06	0.03	0.00	-0.19
39	_red_lcs_lcs	35000	0.02	0.06	0.03	0.00	-0.22
40	_red_lcs_lcs	70000	0.02	0.06	0.03	0.00	-0.19
41	_red_dl_dl	5000	0.02	0.06	0.03	0.00	-0.25
42	_red_dl_dl	10000	0.03	0.07	0.04	0.00	-0.20
43	_red_dl_dl	35000	0.02	0.06	0.03	0.00	-0.23
44	_red_dl_dl	70000	0.02	0.06	0.04	0.00	-0.20
45	_red_jaccard_jaccard	5000	0.02	0.06	0.03	0.00	-0.22
46	_red_jaccard_jaccard	10000	0.02	0.06	0.03	0.00	-0.21
47	_red_jaccard_jaccard	35000	0.02	0.06	0.03	0.00	-0.20
48	_red_jaccard_jaccard	70000	0.03	0.06	0.04	0.00	-0.16
49	_red_qgram_qgram	5000	0.02	0.06	0.03	0.00	-0.17
50	_red_qgram_qgram	10000	0.03	0.06	0.04	0.00	-0.16
51	_red_qgram_qgram	35000	0.03	0.06	0.04	0.00	-0.16
52	_red_qgram_qgram	70000	0.03	0.06	0.04	0.00	-0.19


Batch size
5000
10000
35000
70000
100%
15%
2%
Silhouettes
0.0
−0.1
−0.2
clust
clust_tcs
clust_dl
clust_jac
clust_qg
clust_red
clust_tcs_red
clust_dl_red
clust_jac_red
clust_qg_red
Typ danych

Literatura

[1] Wikipedia - wikipedia, wolna encyklopedia. `http://pl.wikipedia.org/wiki/Wikipedia`. Dostęp: 2015-12-01.

[2] Léon Bottou. Stochastic gradient tricks. Grégoire Montavon, Genevieve B. Orr, Klaus-Robert Müller, redaktorzy, *Neural Networks, Tricks of the Trade, Reloaded*, Lecture Notes in Computer Science (LNCS 7700), strony 430–445. Springer, 2012.

[3] Léon Bottou, Yoshua Bengio. Convergence properties of the k-means algorithms. *Advances in Neural Information Processing Systems 7*, strony 585–592. MIT Press, 1995.

[4] Leonid Boytsov. Indexing methods for approximate dictionary searching: Comparative analysis. *J. Exp. Algorithmics*, 16:1.1:1.1–1.1:1.91, 2011.

[5] R. W. Hamming. Error Detecting and Error Correcting Codes. *Bell System Techincal Journal*, 29:147–160, 1950.

[6] Lawrence Hubert, Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[7] J. Koronacki, J. Ćwik. *Statystyczne systemy uczące się*. Wydawnictwa Naukowo-Techniczne, 2005.

[8] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[9] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[10] Andrew Rosenberg, Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning(EMNLP-CoNLL)*, strony 410–420, 2007.

[11] Peter Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.

[12] D. Sculley. Web-scale k-means clustering. *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, strony 1177–1178, New York, NY, USA, 2010. ACM.

[13] Esko Ukkonen. Algorithms for approximate string matching. *Inf. Control*, 64(1-3):100–118, 1985.

[14] Esko Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92(1):191 – 211, 1992.

[15] Mark P. J. van der Loo. The stringdist Package for Approximate String Matching. *The R Journal*, 6:111–122, 2014.

[16] Robert A. Wagner, Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

[17] Robert A. Wagner, Roy Lowrance. An extension of the string-to-string correction problem. *J. ACM*, 22(2):177–183, 1975.

[18] William E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. *Proceedings of the Section on Survey Research*, strony 354–359, 1990.

[19] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, Dan Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, 2007.