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 [3].*

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 [3]. 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 $s_{i:j} = \varepsilon$ [3, 10].

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 [3].*

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 [3].*

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) [6]. *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 [3].

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 $\mathbb{B}$ [3].

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 [3]:*

- *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 [3].*

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) $\delta(a \to b)$ [11]. 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 $\delta(E) = \sum_{i=1}^{|E|} \delta(a_i \to b_i)$ [3]. 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 [8]. Odległości zdefiniowane w ten sposób zazwyczaj są nazwyane *rozszerzonymi* odległościami edycyjnymi. Przykładowo, $\mathbb{B}$ może zawierać operację `x` $\to$ `ks` o koszcie jednostkowym.

Wówczas rozszerzona odległość pomiędzy napisami `xero` i `ksero` wynosi jeden, podczas gdy standardowa (zwykła, nierozszerzona) odległość wyniosłaby dwa [3].

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 [3].*

[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 [3]:*

- $\delta(s \to t) \in \mathbb{R}$ *(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 [3],*
- *ogólna odległość edycyjna z definicji 1.5 jest metryką [11].*

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* [3].

Lemat 1.4. *Dowolna nieścisła odległość edycyjna ogranicza z dołu odpowiadającą jej ścisłą odległość edycyjną [3].*

Lemat 1.5. *Ścisła odległość Levenshteina o jednostkowym koszcie operacji bazowych jest równa nieścisłej odległości Levenshteina o jednostkowym koszcie operacji bazowych [3].*

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 [3].*

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 [3]. ■

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 [3].

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 [10].

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 [3].

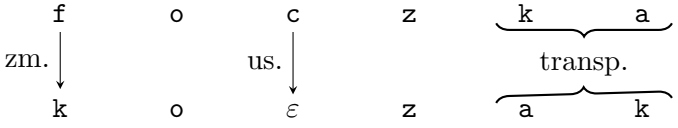
Dopasowanie reprezentuje ścisłą ścieżkę edycyjną $E = s_1 \to t_1, s_2 \to t_2, \dots, s_l \to t_l$. Definiujemy *koszt dopasowania* A jako koszt odpowiadającej mu ścieżki edycyjnej i oznaczamy go przez $\delta(A)$:

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

Optymalne dopasowanie to dopasowanie o najmniejszym koszcie [3].

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 [3].

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 $C_{i,j}$ [3].

Używając równania 1.1 oraz definicji optymalnego dopasowania, łatwo pokazać, że $C_{i,j}$ może zostać policzone przy użyciu następującej ogólnej rekurencji [8]:

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

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,

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

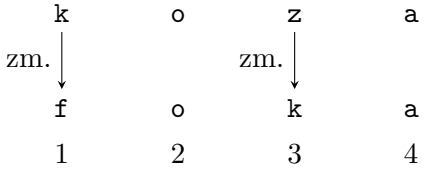
gdzie $[X]$ jest równe jeden, gdy warunek X jest spełniony, zero w przeciwnym przypadku. Co więcej, w tym przypadku możliwa jest tylko jedna kombinacja i' oraz j', mianowicie $i' = i$ oraz $j' = j$. Dalej, odległość ta jest zdefiniowana jedynie dla $|s| = |t|$, zatem $C_{i,j}$ może być policzone jedynie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

Definicja 1.6. Odległością Hamminga *nazywamy [4]:*

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

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!!!!]:

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

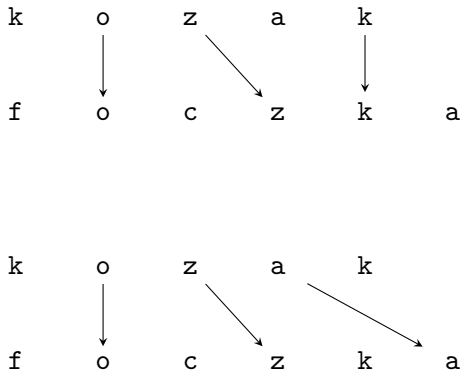
Odległość najdłuższego wspólnego podnapisu przyjmuje wartości ze zbioru $\{0, |s| + |t|\}$, przy czym maksimum jest osiągane, gdy s i t nie mają ani jednego wspólnego znaku. Odległość tę oznaczamy przez d_{lcs}.

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 [10]. 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 [10]. Dobrze obrazuje to następujący przykład:

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

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

bo

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

natomiast

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

bo

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

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

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

bo

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

czyli analogicznie, jak w przypadku 1.6. Natomiast

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

bo

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

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 [3]:

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

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 [12]. 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 [10]. Innymi słowy, do $\mathbb{B}$ należą wstawienia, usunięcia, zamiany oraz operacje $axb \to bya$ o koszcie równym $|x| + |y| + 1$ [3]. 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 [10]:

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

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 poprzez 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:

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

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 [10]:

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] [10]. 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 [9]:*

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

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 [10].

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 [10].

Skoro zdefiniowana została odległość q-gramowa w języku wektorów, każda funkcja odległościowa (distance functions) 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 [9].]

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 [10].

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 [10]:

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:*

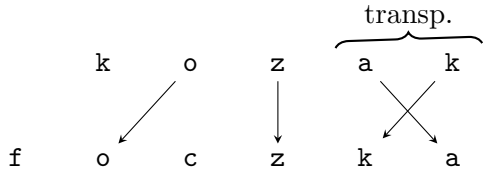
$$|i - j| < \left\lfloor \frac{max\{|s|, |t|\}}{2} \right\rfloor$$

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 poprzez włączenie dodatkowej kary za błędny znak wśród pierwszych czterech znaków napisu [10]:

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 [13]:*

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

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[W LOO JEST BLAD!!]. Powód jest taki, że podobno [PISAC 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 [10]. Wikler [13] 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 $p \in [0, \frac{1}{4}]$ [10].

1.2.4. Podsumowanie

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 [5]. 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 [5]

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

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 [5]:

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

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 [5].

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 [5]

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

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 [14].

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 [2] 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 [1]. 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??] [1]. 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 [2]:

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

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 $\mathbf{x}_i$ [1]:

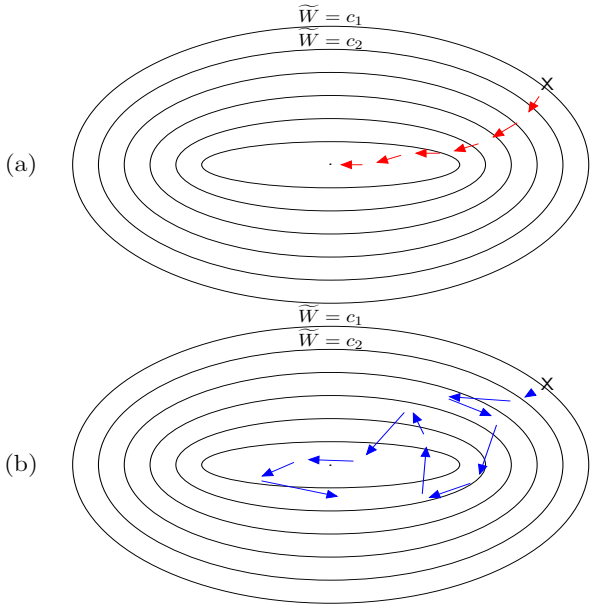
$$\mathbf{m}_l^{(t+1)} = \mathbf{m}_l^{(t)} + \gamma \frac{\partial \widetilde{W}(\mathbf{m}, \mathbf{x}_i)}{\partial \mathbf{m}} \tag{2.7}$$

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 [2]:

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

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

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 [2]. Algorytm ten jest dużo szybszy od dwóch wcześniejszych, kosztem dokładności rozwiązania [1].

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 [7]. 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 [7] O TYM ZE MINI-BATCH JEST TAKI SUPER W POR. Z INNYMI??]

Literatura

[1] 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.

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

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

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

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

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

[7] 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.

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

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

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

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

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

[13] 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.

[14] 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, Grudzie/n 2007.