

Uniwersytet im. Adama Mickiewicza w Poznaniu  
Wydział Matematyki i Informatyki

**Krzysztof Sielski**  
nr albumu: 301650

# **Grupowanie dokumentów tekstowych z wykorzystaniem technik NLP**

Praca magisterska na kierunku:  
informatyka, specjalność: inżynieria oprogramowania

Promotor:  
**dr hab. Krzysztof Jassem**

Poznań, 2010

# Oświadczenie

Poznań, dnia .....

Ja, niżej podpisany Krzysztof Sielski, student Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu, oświadczam, że przedkładaną pracę dyplomową pt.: *Grupowanie dokumentów tekstowych z wykorzystaniem technik NLP* napisałem samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób.

Oświadczam również, że egzemplarz pracy dyplomowej w formie wydruku komputerowego jest zgodny z egzemplarzem pracy dyplomowej w formie elektronicznej.

Jednocześnie przyjmuję do wiadomości, że gdyby powyższe oświadczenie okazało się nieprawdziwe, decyzja o wydaniu mi dyplomu zostanie cofnięta.

.....

---

# Spis treści

---

<b>Spis treści</b>	<b>i</b>
<b>Wstęp</b>	<b>2</b>
<b>1 Modelowanie dokumentów na potrzeby grupowania</b>	<b>4</b>
1.1 Model wektorowy . . . . .	4
1.1.1 Sposoby ważenia terminów w modelu wektorowym . . . . .	5
1.1.2 Funkcje podobieństwa w modelu wektorowym . . . . .	7
1.2 Model grafowy . . . . .	8
1.2.1 Funkcje podobieństwa w modelu grafowym . . . . .	9
<b>2 Algorytmy grupowania</b>	<b>11</b>
2.1 Algorytmy grupowania płaskiego . . . . .	11
2.1.1 Algorytm k-średnich . . . . .	11
2.1.2 Algorytm k-medoidów . . . . .	13
2.1.3 Algorytm rozmyty c-średnich . . . . .	14
2.2 Algorytmy hierarchiczne . . . . .	15
2.2.1 Aglomeracyjne algorytmy hierarchiczne . . . . .	15
2.2.2 Dzielące algorytmy hierarchiczne . . . . .	16
2.3 Algorytmy grupowania oparte na gęstości . . . . .	17
2.4 Algorytmy oparte na frazach . . . . .	18
2.4.1 Algorytm Suffix Tree Clustering . . . . .	18
2.5 Algortmy oparte na ukrytym indeksowaniu semantycznym . . . . .	21
2.5.1 Algorytm Lingo . . . . .	21
<b>3 Metody doskonalenia reprezentacji dokumentów</b>	<b>24</b>
3.1 Pomijanie wyrazów funkcyjnych . . . . .	24
3.2 Lematyzacja . . . . .	25
3.3 Selekcja wyrazów o określonej części mowy . . . . .	25
3.4 Rozpoznawanie fraz . . . . .	26

3.4.1	Rozpoznawanie jednostek nazwanych . . . . .	26
3.5	Semantyka . . . . .	27
3.5.1	Ujednoznacznianie leksemów . . . . .	28
3.5.2	Uwzględnienie relacji semantycznych w reprezentacji wektorowej . . . . .	28
<b>4</b>	<b>Implementacja metod grupowania dokumentów</b>	<b>30</b>
4.1	Reprezentacja dokumentów . . . . .	30
4.1.1	Standardowa reprezentacja dokumentu . . . . .	30
4.1.2	Reprezentacja dokumentu oparta na metodach analizy językowej . . . . .	32
4.1.3	Reprezentacja wektora terminów . . . . .	34
4.2	Zaimplementowane algorytmy grupowania . . . . .	36
4.3	Prezentacja wyników . . . . .	39
<b>5</b>	<b>Badanie efektywności grupowania dokumentów</b>	<b>41</b>
5.1	Ocena reprezentacji dokumentów . . . . .	41
5.1.1	Cel i metoda ewaluacji eksperymentu . . . . .	41
5.1.2	Korpus testowy . . . . .	43
5.1.3	Wyniki eksperymentu i wnioski . . . . .	44
5.2	Ocena algorytmów grupowania . . . . .	49
5.2.1	Metoda ewaluacji eksperymentu . . . . .	49
5.2.2	Opis eksperymentu . . . . .	50
5.2.3	Wyniki eksperymentu i wnioski . . . . .	51
	<b>Podsumowanie</b>	<b>57</b>
	<b>Bibliografia</b>	<b>58</b>
<b>A</b>	<b>Instrukcja użycia aplikacji Ugrupator</b>	<b>61</b>
A.1	Wymagania techniczne . . . . .	61
A.2	Korzystanie z aplikacji . . . . .	61
<b>B</b>	<b>Korpus testowy</b>	<b>63</b>

---

# Podziękowania

---

Swoje podziękowania chciałbym skierować do wszystkich osób, które pomogły mi w pisaniu tej pracy. Mojemu promotorowi, Krzysztofowi Jassemowi, dziękuję za wiele cennych uwag, wskazanie właściwych dróg dla moich działań oraz istotną pomoc w uzyskaniu niezbędnych zasobów do implementacji projektu. Za udostępnienie mi silnika systemu tłumaczenia automatycznego Translatica dziękuję Filipowi Gralińskiemu. Dostęp do plików projektu SłowoSieci zawdzięczam Maciejowi Piaseckiemu. Pragnę również docenić istotne uwagi dotyczące zagadnienia grupowania dokumentów tekstowych od Dawida Weissa. Na końcu chciałbym podziękować swoim przyjaciołom, którzy poświęcili swój czas na udział w przeprowadzonych przeze mnie badaniach.

---

# Wstęp

---

Istotą **grupowania** (zwanego również **analizą skupień** lub **klasteryzacją**, ang. *data clustering*) jest podział wejściowego zbioru elementów na grupy (inaczej: **klastry**) w taki sposób, by elementy przypisane do jednej grupy były jak najbardziej do siebie podobne oraz by to podobieństwo było minimalne dla elementów z różnych grup. Analiza skupień ma wiele różnych zastosowań, można ją wykorzystać na przykład do analizy obrazów czy identyfikowania typów klientów w celach marketingowych. Niniejsza praca prezentuje możliwości jej wykorzystania do grupowania dokumentów tekstowych.

W jakim celu grupuje się dokumenty? Najczęściej powtarzaną odpowiedzią na to pytanie jest **hipoteza klastrowania** (ang. *cluster hypothesis*), która mówi, że dokumenty należące do tego samego klastra odpowiadają na tę samą potrzebę informacji<sup>1</sup>. Prawdopodobnie najbardziej znanym przykładem zastosowania tego zagadnienia jest serwis Google News<sup>2</sup>, który grupuje artykuły dotyczące tego samego wydarzenia z wielu serwisów informacyjnych, umożliwiając zapoznanie się z różnymi punktami widzenia na ten sam temat. Każda z tych grup dokumentów zostaje przydzielona do jednej z predefiniowanych kategorii, takich jak *Sport*, *Rozrywka* czy *Gospodarka*. W tym ostatnim przypadku mamy do czynienia z **klasyfikacją**, która jest często mylona z klasteryzacją. Istotnie, są to zbliżone pojęcia, istnieją jednak między nimi wyraźne różnice. Proces klasyfikacji polega na przypisaniu dokumentu do jednej ze zdefiniowanych uprzednio kategorii i jest przykładem metody **nauczania nadzorowanego** (ang. *supervised learning*), gdyż wymaga stworzenia przez człowieka listy kategorii i zbioru treningowego (tj. przykładowych dokumentów należących do każdej z kategorii). Natomiast w trakcie klasteryzacji odpowiednie kategorie są tworzone na podstawie wejściowych danych bez ingerencji ludzkiej, dlatego ta metoda jest czasami zwana klasyfikacją bez nadzoru (**uczenie nienadzorowane**, ang. *unsupervised*

---

<sup>1</sup>Jest to wolny przekład sformułowania *Documents in the same cluster behave similarly with respect to relevance to information needs*, którym opisana jest hipoteza klastrowania w książce [MRS08]

<sup>2</sup><http://news.google.pl/>

*learning*).

Innym pojęciem o znaczeniu zbliżonym do grupowania dokumentów jest **detekcja niemal-duplikatów** (ang. *near-duplicate detection*). Proces ten ma na celu znalezienie niemal identycznych dokumentów tekstowych (różniących się na przykład tylko tytułem nagłówka), między którymi zachodzi dużo większe podobieństwo niż ma to zwykle miejsce w przypadku użycia metod analizy skupień.

Gdy mówimy o grupowaniu podobnych dokumentów, należałoby zdefiniować pojęcie *podobieństwa*. Istnieje wiele różnych formalizacji tego pojęcia i zostaną one podane w następnych rozdziałach. W niniejszej pracy najważniejsze i najczęściej używane będzie jednak intuicyjne rozumienie tego słowa. Ocena, czy dwa dokumenty są podobne, pozostanie zawsze subiektywna i zależna od kontekstu: czasem wystarczy ta sama tematyka (np. sport, polityka), czasem zaś wymagamy od tekstów opisywania tego samego wydarzenia (np. powódź we Wrocławiu w maju 2010 roku).

## Struktura pracy

Niniejsza praca składa się z pięciu rozdziałów. Rozdział pierwszy opisuje sposoby sprowadzania tekstów do formy umożliwiającej ich dalszą analizę i porównywanie. Rozdział drugi prezentuje różne algorytmy grupowania dokumentów tekstowych. W rozdziale trzecim zaproponowane są metody ulepszenia reprezentacji dokumentów tak, by lepiej odzwierciedlały ich tematykę i możliwe było trafniejsze ich porównywanie. Czwarty rozdział opisuje techniki i algorytmy zaimplementowane w autorskim programie do grupowania tekstów. Skuteczność jego działania została zbadana w szeregu eksperymentów, których przebieg i metody ewaluacji zostały opisane w piątym rozdziale wraz z wnioskami z nich wyciągniętymi.

## Rozdział 1

---

# Modelowanie dokumentów na potrzeby grupowania

---

Aby móc grupować dokumenty tekstowe, niezbędne jest określenie sposobu ich porównywania. Niniejszy rozdział opisuje metody formalnej reprezentacji tekstów, które umożliwiają obliczanie podobieństwa pomiędzy nimi.

### 1.1 Model wektorowy

Najbardziej rozpowszechnionym sposobem reprezentowania dokumentów tekstowych w algorytmach grupowania jest **model wektorowy** (ang. *Vector Space Model*), który do modelowania tekstów wykorzystuje pojęcia algebry liniowej. Został on zaproponowany w artykule [SWY75]. W modelu tym dokument zapisywany jest jako wektor liczb rzeczywistych  $[w_{t_1}, w_{t_2}, \dots, w_{t_n}]$ , gdzie  $w_{t_i}$  oznacza wagę **terminu**<sup>1</sup> $t_i$ , zaś  $\Omega = \{t_1, t_2, \dots, t_n\}$  jest zbiorem wszystkich terminów. Sposoby ważenia terminów zostaną dokładniej omówione w rozdziale 1.1.1, do tego czasu przyjmijmy założenie, że waga terminu  $t$  w dokumencie  $d$  będzie wynosić 1, gdy termin  $t$  występuje w treści dokumentu  $d$  i 0 w przeciwnym przypadku. W niniejszej pracy reprezentacja wektorowa dokumentu  $d$  oznaczana będzie symbolem  $\vec{d}$ .

Zgodnie z modelem wektorowym,  $m$ -elementowy korpus (zbiór  $m$  dokumentów tekstowych) reprezentowany jest przez macierz  $C_{m \times n}$  (ang. *term-document matrix*), w której kolejne wiersze są wektorami odpowiadającymi poszczególnym dokumentom. Przykład 1 prezentuje konstrukcję takiej macierzy.

Zauważmy, że model wektorowy traktuje dokument jako multizbiór słów (ang. *bag of words*). Pomijane są zatem informacje o strukturze tekstu (podział na na-

---

<sup>1</sup>*termin* może oznaczać pojedyncze słowo lub ciąg słów, np. imię i nazwisko



Dokument	Treść
$d_1$	Recesja mięknie, ale bezrobocie straszy
$d_2$	Informatyczna recesja
$d_3$	Koniec najdłuższej recesji od czasu Wielkiego Kryzysu

$\Omega = (\text{ale, bezrobocie, czasu, informatyczna, koniec, mięknie, najdłuższej, od, recesja, recesji, straszy, Wielkiego Kryzysu})$

$$\vec{d}_1 = (1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0)$$

$$\vec{d}_2 = (0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0)$$

$$\vec{d}_3 = (0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1)$$

$$C = \begin{pmatrix} \vec{d}_1 \\ \vec{d}_2 \\ \vec{d}_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

**Przykład 1:** Reprezentacja wektorowa dokumentów;  $\Omega$  — zbiór terminów występujących w dokumentach;  $C$  — reprezentacja macierzowa korpusu

główki, tytuły, akapity), a także o kolejności słów. Ta ostatnia ma duże znaczenie w takich językach jak angielski, gdzie semantyka zdania zależy od kolejności słów: dla przykładu *John loves Mary* i *Mary loves John* mają identyczną reprezentację wektorową, mimo że ich znaczenie jest różne.

Prostota oraz duża szybkość obliczeń wektorowych sprawia, iż model ten stał się najpopularniejszym sposobem reprezentowania dokumentów.

### 1.1.1 Sposoby ważenia terminów w modelu wektorowym

Zgodnie z tym, co zostało wspomniane powyżej, w celu zapisania dokumentu w postaci wektorowej należy wybrać metodę obliczania wagi terminu  $t$  w dokumencie  $d$ . **Wagą** będziemy nazywać funkcję przyporządkowującą danemu terminowi w danym dokumencie nieujemną liczbę rzeczywistą:  $w_C : \Omega \times C \rightarrow \langle 0, \infty \rangle$ . Waga jest funkcją zdefiniowaną na danym korpusie (jej sposób obliczania jest zależny od korpusu, m.in. od liczby dokumentów).

Jedną z najprostszych funkcji ważących jest **częstość występowania** (ang. *term frequency*), tradycyjnie oznaczana w literaturze jako  $w_{tf}(t, d)$ , oznaczająca liczbę wystąpień terminu  $t$  w dokumencie  $d$ . Zauważmy, iż terminy popularne w całym korpusie (takie jak spójniki: *i*, *lub*), które nie posiadają właściwości dystynktywnych, posiadałyby wysoką wagę  $w_{tf}$  w każdym (lub prawie każdym) dokumencie. Taka cecha nie jest pożądana, dlatego w praktyce funkcja ta jest stosowana rzadko. Wagę, która zależy jedynie od zawartości dokumentu i jest niezależna od własności innych dokumentów w korpusie, będziemy nazywać **wagą lokalną**.

Wagę, której wartość zależy od wszystkich dokumentów w korpusie, nazywać będziemy **wagą globalną**. Przykładem takiej wagi jest **odwrotna częstość dokumentowa** (ang. *inverse document frequency*), oznaczana jako  $w_{idf}(t, d)$  i definiowana w następujący sposób:

$$w_{idf}(t, d) = \log \frac{|C|}{|c \in C : t \in c|}$$

gdzie:  $|C|$  oznacza liczbę dokumentów w korpusie, zaś  $|c \in C : t \in c|$  oznacza liczbę dokumentów w korpusie zawierających termin  $t$ .

Jak łatwo zauważyć, waga  $w_{idf}$  nie zależy od dokumentu, dla którego jest liczona, a jedynie od częstości występowania danego terminu w korpusie, przyjmując wartości wysokie dla terminów rzadkich, a niskie — dla częstych.

Z dwóch opisanych powyżej funkcji składa się waga  $w_{tf-idf}$ , która łączy zalety  $w_{tf}$  i  $w_{idf}$ . Obliczana jest ona w następujący sposób:

$$w_{tf-idf}(t, d) = w_{tf}(t, d) \cdot w_{idf}(t, d) = w_{tf}(t, d) \cdot \log \frac{|C|}{|c \in C : t \in c|}$$

Funkcja  $w_{tf-idf}$  przyjmuje wysokie wartości dla terminów, które występują często w małej liczbie dokumentów, średnie dla terminów pojawiających się w wielu dokumentach lub pojawiających się rzadko w niewielu dokumentach i niskie dla terminów występujących we wszystkich (lub prawie wszystkich) dokumentach. Własności te są bardzo pożądane z punktu widzenia aplikacji grupującej dokumenty, gdyż umożliwiają rozpoznanie terminów charakterystycznych. Z tego powodu waga ta jest często wykorzystywana w aplikacjach, które stosują model wektorowy.

Czasami stosowane są również inne, bardziej skomplikowane wagi. Jedną z nich jest wywodząca się z teorii informacji waga **Pointwise Mutual Information** (PMI). W ogólności, PMI jest sposobem mierzenia współzależności zmiennych losowych dyskretnych:  $PMI(x, y) = \log_2 \frac{p(x, y)}{p(x) \cdot p(y)}$ . W odniesieniu do dokumentów tekstowych waga  $w_{pmi}$  ma postać:

$$w_{pmi}(t, d) = \log_2 \frac{\frac{w_{tf}(t, d)}{|C|}}{\frac{\sum_{i \in \Omega} w_{tf}(i, d)}{|C|} \cdot \frac{\sum_{j \in C} w_{tf}(t, j)}{|C|}}$$

Wektory uzyskane poprzez ważenie terminów zgodnie z opisanymi wcześniej funkcjami mogą mieć różną długość (rozumianą jako norma euklidesowa). Wagi zależne są zwykle od liczby wystąpień danego terminu w dokumencie, więc reprezentacje dłuższych tekstów mają większą wartość normy euklidesowej. Aby wyeliminować te różnice, często wektory normalizuje się — po to, by miały długość jednostkową. Normalizacja polega na podzieleniu wartości każdego współczynnika wektora  $\vec{d}$  przez jego normę  $\|\vec{d}\|_2$ . Wektor jednostkowy  $\hat{d}$  obliczamy

w następujący sposób:

$$\hat{d} = \frac{\vec{d}}{\|\vec{d}\|_2} = \frac{\vec{d}}{\sqrt{\sum_{i=1}^n w_i^2}}$$

### 1.1.2 Funkcje podobieństwa w modelu wektorowym

Podobieństwo<sup>2</sup> między dokumentami reprezentowanymi przez wektory obliczane jest funkcją zdefiniowaną następująco:

$$sim : C \times C \rightarrow [0, 1]$$

Funkcja ta powinna być symetryczna i spełniać następujące dwa warunki:

- jeżeli wektory dokumentów są takie same, to wartość podobieństwa wynosi 1, tj:

$$\forall_{i \in \{1..n\}} w_{1i} = w_{2i} \Rightarrow sim(d_1, d_2) = 1$$

- jeżeli nie istnieje termin, dla którego wektory obu dokumentów mają dodatnią wagę, to wartość podobieństwa wynosi 0:

$$\forall_{i \in \{1..n\}} w_{1i} \cdot w_{2i} = 0 \Rightarrow sim(d_1, d_2) = 0$$

Jeżeli dokumenty posiadają znormalizowane reprezentacje wektorowe, do obliczania podobieństwa pomiędzy nimi możemy wykorzystać metrykę euklidesową (w przypadku nieznormalizowanych wektorów metryka ta będzie dawać zaburzone wyniki). Odległość euklidesową wektorów obliczamy w następujący sposób:

$$dist_{euclid}(\vec{d}_1, \vec{d}_2) = \sqrt{\sum_{i=1}^n (w_{1j} - w_{2j})^2} \quad (1.1)$$

gdzie:  $\vec{d}_1 = [w_{11}, w_{12}, \dots, w_{1n}]$ ,  $\vec{d}_2 = [w_{21}, w_{22}, \dots, w_{2n}]$ .

Zauważmy, że jeżeli  $\vec{d}_1$  i  $\vec{d}_2$  są wektorami jednostkowymi posiadającymi nieujemne współczynniki, to

$$\forall_{\vec{d}_1, \vec{d}_2} dist_{euclid}(\vec{d}_1, \vec{d}_2) \in [0, \sqrt{2}]$$

Z uwagi na powyższe oraz fakt, iż podobieństwo między dokumentami jest tym większe, im mniejsza jest odległość między ich reprezentacjami wektorowymi, podobieństwo możemy obliczyć w następujący sposób:

$$sim_{euclid}(d_1, d_2) = \frac{\sqrt{2} - dist_{euclid}(\vec{d}_1, \vec{d}_2)}{\sqrt{2}} = \frac{\sqrt{2} - \sqrt{\sum_{i=1}^n (w_{1j} - w_{2j})^2}}{\sqrt{2}}$$

<sup>2</sup>Zamiast definiowania miary podobieństwa między dokumentami, niektórzy definiują miarę odległości pomiędzy nimi

Innym, nie wymagającym normalizacji wektorów reprezentacji dokumentów sposobem obliczania podobieństwa, jest użycie kosinusa kąta pomiędzy wektorami:

$$\text{sim}_{\text{cosine}}(d_1, d_2) = \frac{\vec{d}_1 \cdot \vec{d}_2^T}{\|\vec{d}_1\|_2 \cdot \|\vec{d}_2\|_2} = \frac{\sum_{i=1}^n (w_{1i} \cdot w_{2i})}{\sqrt{\sum_{i=1}^n w_{1i}^2 \cdot \sum_{i=1}^n w_{2i}^2}} \quad (1.2)$$

Jeżeli ponadto wektory  $\vec{d}_1$  i  $\vec{d}_2$  we wzorze 1.2 będą znormalizowane, to mianownik w tym wzorze będzie mieć wartość 1, zatem może zostać pominięty:

$$\text{sim}_{\text{cosine}}(d_1, d_2) = \vec{d}_1 \cdot \vec{d}_2^T = \sum_{i=1}^n w_{1i} \cdot w_{2i} \quad (1.3)$$

## 1.2 Model grafowy

Model wektorowy jest najpopularniejszym sposobem reprezentacji dokumentów, nie jest on jednak jedyny. Podejmowane były próby modelowania dokumentów za pomocą grafów, które — w przeciwieństwie do wektorów — zachowują informacje o strukturze tekstu i kolejności wyrazów w zdaniach. Kilka sposobów grafowej reprezentacji dokumentów zaproponowanych jest w artykule [SBLK05]. Są to następujące metody:

**uproszczona** (ang. *simple*) — każdy termin  $t$  pojawiający się w dokumencie  $d$  reprezentowany jest jako wierzchołek  $v_t$  w grafie skierowanym  $G_d$ . Jeżeli terminy  $t_i$  i  $t_j$  wystąpiły w tym samym zdaniu w dokumencie  $d$  oraz  $t_i$  bezpośrednio poprzedza wystąpienie terminu  $t_j$ , to w grafie  $G_d$  istnieje krawędź skierowana z wierzchołka  $v_{t_i}$  do wierzchołka  $v_{t_j}$ .

**standardowa** (ang. *standard*) — różni się tym od uproszczonej, iż krawędziom zostaje przypisana etykieta opisująca miejsce w dokumencie, gdzie dana para terminów się pojawiła (np. krawędź łącząca dwa wierzchołki reprezentujące terminy występujące w tytule otrzyma etykietę *tytuł*, a łącząca wierzchołki reprezentujące terminy występujące w treści paragrafu — etykietę *paragraf*, itd.)

**uproszczona  $n$ -odległości** (ang.  *$n$ -simple distance*) — jest podobna do uproszczonej, lecz pod uwagę branych jest  $n$  terminów występujących po danym.

**$n$ -odległości** (ang.  *$n$ -distance*) — uzyskiwana jest z uproszczonej  $n$ -odległości poprzez etykietowanie krawędzi odległością pomiędzy danymi terminami (tzn. krawędzie łączące terminy bezpośrednio po sobie występujące otrzymują etykietę 1, terminy rozdzielone innym terminem: 2, itd.)

**częstotliwości bezwzględnej** (ang. *absolute frequency*) — do reprezentacji uproszczonej dodawane są etykiety: wierzchołki etykietowane są liczbą wystąpień w danym dokumencie terminu reprezentowanego przez ów wierzchołek, zaś krawędzie — liczbą wystąpień pary terminów reprezentowanej przez tę krawędź

**częstotliwości względnej** (ang. *relative frequency*) — stanowi udoskonalenie reprezentacji częstotliwości bezwzględnej — wartości etykiet dla wierzchołków i krawędzi są znormalizowane tak, by długie i krótkie dokumenty tekstowe mogły być porównywane. Normalizacja polega na podzieleniu każdej wartości etykiety wierzchołka przez największą wartość etykiety dla wierzchołka, analogicznie postępuje się w stosunku do krawędzi.

### 1.2.1 Funkcje podobieństwa w modelu grafowym

W celu omówienia funkcji podobieństwa w modelu grafowym, konieczne jest wprowadzenie dwóch pojęć: *maksymalnego wspólnego podgrafu* i *minimalnego wspólnego nadgrafu*.

**Największym wspólnym podgrafem** grafów  $G_{d_1}$  i  $G_{d_2}$  będziemy nazywać graf  $mcs(G_{d_1}, G_{d_2})$  spełniający poniższą zależność:

$$\begin{aligned} mcs(G_{d_1}, G_{d_2}) = G_{mcs} &\Leftrightarrow G_{mcs} \subseteq G_{d_1} \wedge G_{mcs} \subseteq G_{d_2} \\ &\wedge \neg \exists G' \neq G_{mcs} (G' \subseteq G_{d_1} \wedge G' \subseteq G_{d_2} \wedge |G'| > |G_{mcs}|) \end{aligned} \quad (1.4)$$

przy czym  $|G|$  oznacza wielkość grafu  $G$ , liczoną jako sumę liczby wierzchołków i krawędzi tego grafu<sup>3</sup>.

**Największym wspólnym nadgrafem** grafów  $G_{d_1}$  i  $G_{d_2}$  będziemy nazywać graf  $MCS(G_{d_1}, G_{d_2})$  spełniający poniższą zależność:

$$\begin{aligned} MCS(G_{d_1}, G_{d_2}) = G_{MCS} &\Leftrightarrow G_{d_1} \subseteq G_{MCS} \wedge G_{d_2} \subseteq G_{MCS} \\ &\wedge \neg \exists G' \neq G_{MCS} (G_{d_1} \subseteq G' \wedge G_{d_2} \subseteq G' \wedge |G'| < |G_{MCS}|) \end{aligned} \quad (1.5)$$

Na powyżej zdefiniowanych pojęciach oparte są następujące miary podobieństwa dokumentów reprezentowanych przez grafy  $G_{d_1}$  i  $G_{d_2}$  (za [SBLK05]):

$$sim_{mcs}(d_1, d_2) = \frac{|mcs(G_{d_1}, G_{d_2})|}{\max(|G_{d_1}|, |G_{d_2}|)} \quad (1.6)$$

$$sim_{WGU}(d_1, d_2) = \frac{|mcs(G_{d_1}, G_{d_2})|}{|G_{d_1}| + |G_{d_2}| - |mcs(G_{d_1}, G_{d_2})|} \quad (1.7)$$

$$sim_{MMSCN}(d_1, d_2) = \frac{|mcs(G_{d_1}, G_{d_2})|}{|MCS(G_{d_1}, G_{d_2})|} \quad (1.8)$$

<sup>3</sup>w teorii grafów wielkość grafu oznacza zwykle liczbę jego wierzchołków

Inny sposób mierzenia podobieństwa opiera się na odległości edycyjnej między grafami. **Odległością edycyjną** między grafami  $G_{d_1}$  i  $G_{d_2}$  nazywamy liczbę elementarnych operacji, jakie należy wykonać, aby przekształcić graf  $G_{d_1}$  w graf  $G_{d_2}$ . Oznaczając będziemy ją  $dist_{edit}(G_{d_1}, G_{d_2})$ . Do operacji elementarnych zaliczamy usunięcie, wstawienie albo zamianę wierzchołka lub krawędzi grafu. Miarę podobieństwa wyprowadzamy z odległości edycyjnej w następujący sposób:

$$sim_{edit}(d_1, d_2) = 1 - \frac{dist_{edit}(G_{d_1}, G_{d_2})}{|G_{d_1}| + |G_{d_2}|} \quad (1.9)$$

## Rozdział 2

---

# Algorytmy grupowania

---

Niniejszy rozdział stanowi przegląd różnych metod używanych w analizie skupień. Większość z przedstawionych tutaj algorytmów ma zastosowanie dla różnego rodzaju danych, nie tylko dokumentów tekstowych. Różnią się one między sobą nie tylko mechanizmem działania, ale i strukturą wynikowego podziału danych na grupy. Jak zostanie pokazane w dalszej części, efektem ich działania może być **podział twardy** (ang. *hard clustering*), w którym każdy dokument należy dokładnie do jednej grupy, lub **podział miękki** (ang. *soft clustering*), w którym dokument może zostać przypisany do kilku grup lub nie należeć do żadnej. Wynikiem grupowania może być również hierarchia grup lub pojedynczy, płaski podział. Większość opisanych algorytmów zakłada wektorową reprezentację danych.

### 2.1 Algorytmy grupowania płaskiego

Algorytmy grupowania płaskiego charakteryzują się tym, że dzielą cały zbiór dokumentów na pewną z góry znaną liczbę klastrów bez wskazywania jakichkolwiek relacji między tymi klastrami.

#### 2.1.1 Algorytm k-średnich

Algorytm k-średnich (ang. *k-means*) jest klasycznym, jednym z najstarszych algorytmów grupowania danych. Po raz pierwszy został zaprezentowany w pracy [Mac67]. Zyskał dużą popularność z uwagi na swoją wyjątkową prostotę oraz niską złożoność obliczeniową.

Algorytm ten działa zgodnie z poniższym schematem:

1. dokumenty zostają przypisane do  $k$  grup w sposób losowy

2. dla każdej grupy  $g$  obliczany jest **centroid**, czyli wektor będący średnią arytmetyczną wektorów modelujących dokumenty należące do tej grupy:

$$\vec{\mu}_g = \frac{1}{|g|} \sum_{\vec{d} \in g} \vec{d} \quad (2.1)$$

3. dokumenty są przypisywane do grupy, której centroid jest najbardziej podobny do danego dokumentu
4. kroki 2 i 3 są powtarzane tak długo, aż przydział dokumentów do poszczególnych grup przestanie się zmieniać (można przyjąć również inne kryterium stopu, np. wykonanie stałej liczby iteracji).

---

**Algorytm 1** Algorytm k-średnich
 

---

**Wejście:** liczba klastrów  $k$ , zbiór dokumentów  $C = \{d_1, d_2, \dots, d_m\}$

**Wyjście:** podział  $G = \{g_1, g_2, \dots, g_k\}$

*// losowe przypisanie dokumentów do grup*

**for all**  $d \in C$  **do**

$i \leftarrow \text{RANDOM}(k)$

$g_i \leftarrow g_i \cup \{d\}$

**end for**

**repeat**

*// obliczenie centroidu każdej grupy*

**for**  $i = 1$  to  $k$  **do**

$\mu_{g_i} \leftarrow \frac{1}{|g_i|} \sum_{\vec{d} \in g_i} \vec{d}$

$g_i \leftarrow \{\}$

**end for**

*// przypisanie dokumentów do najbliższego centroidu*

**for all**  $d \in C$  **do**

$i \leftarrow \arg \max_j \text{sim}(d, \mu_{g_j})$

$g_i \leftarrow g_i \cup \{d\}$

**end for**

**until** podział  $\{g_1, g_2, \dots, g_k\}$  wystąpił po raz drugi

**return**  $\{g_1, g_2, \dots, g_k\}$

---

Istotą działania algorytmu k-średnich jest znalezienie podziału  $G$ , w którym dokumenty są najbardziej podobne do centroidów grup, do których należą, czyli wartość poniższego wyrażenia jest największa:

$$\sum_{g \in G} \sum_{d \in g} \text{sim}(d, \mu_g) \quad (2.2)$$

Niestety, algorytm ten znajduje minima lokalne (nie globalne) powyższej funkcji, dlatego rozwiązanie znalezione przez algorytm k-średnich często nie jest optymalne. Ponadto, algorytm ten nie jest deterministyczny — jego wynik może być



różny dla tych samych danych wejściowych w zależności od początkowego losowego przypisania dokumentów do grup. Na rysunku 1 przedstawiony jest przypadek, gdy algorytm  $k$ -średnich (dla  $k = 2$ ) wskutek niesprzyjającego początkowego losowego przypisania dokumentów do grup zwrócił niepoprawny wynik — znalezione przez niego grupy to  $\{d_1, d_2, d_3\}$  i  $\{d_4, d_5\}$ , podczas gdy prawidłowy podział powinien być następujący:  $\{d_1, d_2, d_3, d_4\}$  i  $\{d_5\}$ . Pewnym sposobem na polepszenie jakości grupowania jest kilkukrotne wykonanie tego algorytmu (z różnymi początkowymi przypisaniami) i wybranie najlepszego wyniku (mającego najwyższą wartość wyrażenia 2.2).

---

**Rysunek 1** Błędne grupowanie zwrócone przez algorytm  $k$ -średnich

---



Dodatkową przeszkodą w użyciu algorytmu  $k$ -średnich jest konieczność przyjęcia a priori, na ile grup analizowana kolekcja dokumentów powinna zostać podzielona. Jednak mimo opisanych wad, algorytm  $k$ -średnich wciąż jest popularny. Powodem tego jest jego niska złożoność obliczeniowa, wynosząca  $O(kmn)$  — czas jego działania jest liniowo zależny od pożądanej liczby klastrów  $k$ , liczby dokumentów  $m$  oraz liczby wszystkich terminów występujących w dokumentach  $n$ . Uzasadnić to można w następujący sposób (za [MRS08]): obliczenie podobieństwa  $sim_{cosine}$  wymaga  $O(n)$  operacji. W głównej pętli następuje obliczenie centroidów (operacja ta ma złożoność  $O(nm)$ ) oraz dla każdego dokumentu obliczane jest jego podobieństwo do każdego centroidu (krok ten składa się z  $O(kmn)$  instrukcji). Jedna iteracja ma zatem złożoność  $O(kmn)$ . Główna pętla może być wprawdzie wykonywana nawet  $\binom{n}{k}$  razy (aż do momentu wyczerpania wszystkich możliwych podziałów), lecz w praktyce algorytm ten jest dość szybko zbieżny do ostatecznego wyniku i najczęściej stosuje się stałą liczbę iteracji, co nie wpływa na klasę złożoności.

### 2.1.2 Algorytm $k$ -medoidów

Modyfikacją algorytmu  $k$ -średnich jest algorytm  $k$ -medoidów, który pojęcie centroidu grupy zastępuje pojęciem **medoidu** — obiektu istniejącego w kolekcji, którego suma podobieństw do pozostałych dokumentów w grupie jest największa.

Algorytm  $k$ -medoidów ma kilka wersji, jedną z nich jest **PAM** (*Partitioning Around Medoids*). Przebiega on według następującego schematu:

1.  $k$  elementów jest losowanych jako medoidy
2. każdy dokument jest przypisywany do najbliższego medoidu
3. jeżeli zastąpienie któregoś medoidu pewnym dokumentem (nie będącym medoidem) powoduje zwiększenie wartości wyrażenia

$$\sum_{g \in G} \sum_{d \in g} sim(d, m_g)$$

( $m_g$  — medoid grupy  $g$ ), to taka zamiana zostaje wykonana

4. kroki 2 i 3 są powtarzane tak długo, aż zbiór medoidów przestanie się zmieniać

Algorytm k-medoidów w porównaniu z algorytmem k-średnich jest bardziej odporny na istnienie *punktów osobliwych* w zbiorze danych, czyli dokumentów niepodobnych do pozostałych [For06].

### 2.1.3 Algorytm rozmyty c-średnich

Algorytm rozmyty c-średnich (ang. *fuzzy c-means*) jest podobny do algorytmu k-średnich, opiera się jednak na logice rozmytej, a nie klasycznej, dlatego wynikowy podział dokumentów na grupy jest również nieostry. Przynależność dokumentu do danej grupy wyrażana jest liczbą z zakresu  $[0, 1]$  tak, by suma przynależności dokumentu do wszystkich grup wynosiła 1. Wynikiem działania tego algorytmu jest macierz  $G_{m \times k}$ , w której wartość  $g_{i,j}$  wyraża stopień przynależności dokumentu  $d_i$  do grupy  $j$ .

Poszczególne kroki algorytmu rozmytego c-średnich są analogiczne do kroków algorytmu k-średnich:

1. Macierz  $G_{m \times k}$  jest inicjalizowana losowymi wartościami z zakresu  $[0, 1]$  tak, by zachowany był poniższy warunek:

$$\forall_{i=1}^m : \left( \sum_{j=1}^k g_{i,j} \right) = 1$$

2. Dla każdej grupy ( $j = 1, \dots, k$ ) obliczany jest centroid:

$$\vec{\mu}_j = \frac{\sum_{i=1}^m g_{i,j}^n \cdot \vec{d}_i}{\sum_{i=1}^m g_{i,j}^n}, n \in (1, \infty)$$

W tym przypadku centroid grupy  $j$  jest średnią wszystkich dokumentów z korpusu, ważoną stopniem przynależności danego dokumentu do  $j$ -tej grupy.

3. Obliczana jest macierz  $G$  zgodnie z następującym wzorem:

$$g(i, j) = \frac{1}{\sum_{a=1}^k \left( \frac{\text{sim}(\mu_j, d_i)}{\text{sim}(\mu_a, d_i)} \right)^{\frac{2}{m-1}}}$$

4. Kroki 2 i 3 są powtarzane tak długo, aż wszystkie współczynniki macierzy przynależności otrzymanej w dwóch kolejnych krokach będą się różnić o mniej niż wcześniej wybrana wartość  $\epsilon \in (0, 1]$ :

$$\|G(i+1) - G(i)\|_{\max} < \epsilon$$

gdzie:  $\|G\|_{\max} = \max\{|g_{i,j}|\}$  — norma maksimum,  $G(i)$  — macierz przynależności otrzymana w  $i$ -tej iteracji algorytmu.

Wyniki uzyskiwane w algorytmie rozmytym c-średnich są jakościowo zbliżone do tych uzyskiwanych algorytmem k-średnich, różnią się jednak formą — dokumenty mogą należeć do kilku różnych grup w różnym stopniu, co może być cechą pożądaną w niektórych zastosowaniach.

## 2.2 Algorytmy hierarchiczne

Algorytmy hierarchiczne tworzą hierarchię podziałów dokumentów na grupy, umożliwiając obserwację wyniku na różnych poziomach szczegółowości. Wynikiem działania tych algorytmów jest drzewo binarne, w którym węzłami są grupy dokumentów, przy czym korzeniem jest grupa zawierająca wszystkie dokumenty, a liśćmi — grupy jednoelementowe.

Algorytmy tego rodzaju przebiegają zgodnie z jednym z poniższych schematów:

**aglomeracyjne algorytmy hierarchiczne** (ang. *hierarchical agglomerative clustering*, HAC) zaczynają od stworzenia podziału, w którym każdy dokument stanowi odrębną jednoelementową grupę i w każdym kolejnym kroku dwie najbardziej do siebie podobne grupy zostają połączone

**dzielące algorytmy hierarchiczne** (ang. *hierarchical divisive clustering*, HDC) tworzą na początku grupę zawierającą wszystkie dokumenty z kolekcji i w kolejnych krokach rekurencyjnie dzielą jedną z grup na dwie podgrupy.

### 2.2.1 Aglomeracyjne algorytmy hierarchiczne

Poszczególne algorytmy hierarchicznego grupowania aglomeracyjnego w różny sposób realizują łączenie najbardziej podobnych do siebie grup. Miara podobieństwa pomiędzy grupami może być zdefiniowana w różny sposób, a najczęściej stosowane miary to:

**single-linkage** — podobieństwo pomiędzy grupami jest równe wartości podobieństwa dwóch najbliższych sobie elementów z tych grup:

$$SIM_{single-linkage}(g_1, g_2) = \max_{d_1 \in g_1, d_2 \in g_2} \{sim(d_1, d_2)\}$$

**complete-linkage** — podobieństwo między grupami równe jest wartości podobieństwa dwóch najbardziej od siebie odległych elementów tych grup:

$$SIM_{complete-linkage}(g_1, g_2) = \min_{d_1 \in g_1, d_2 \in g_2} \{sim(d_1, d_2)\}$$

**centroid** — (*Unweighted Pairwise Group Method with Centroids, UPGMC*) podobieństwo między grupami równe jest wartości podobieństwa ich centroidów

$$SIM_{centroid}(g_1, g_2) = sim(\mu_{g_1}, \mu_{g_2})$$

**arithmetic-mean** — podobieństwo między grupami równe jest średniej arytmetycznej wartości podobieństwa pomiędzy każdym dokumentem z pierwszej grupy a każdym dokumentem z drugiej grupy

$$SIM_{arithmetic-mean}(g_1, g_2) = \frac{\sum_{d_1 \in g_1, d_2 \in g_2} sim(d_1, d_2)}{|g_1||g_2|}$$

**group-average** — (*Unweighted Pairwise Group Method with Averages, UPGMA*) podobieństwo między grupami równe jest średniej wartości podobieństwa pomiędzy każdą parą różnych dokumentów należących do obu grup (łącznie z parami dokumentów z tej samej grupy)

$$SIM_{group-average}(g_1, g_2) = \frac{\sum_{d_1 \in g_1 \cup g_2, d_2 \in g_1 \cup g_2, d_1 \neq d_2} sim(d_1, d_2)}{(|g_1| + |g_2|)(|g_1| + |g_2| - 1)}$$

Wszystkie powyżej opisane warianty algorytmu aglomeracyjnego mają złożoność obliczeniową równą  $O(m^2 \log m)$  poza  $SIM_{single-linkage}$ , którego złożoność wynosi  $O(m^2)$  [MRS08]. Seria eksperymentów opisana w [SKK00] wykazała, że najlepsze rezultaty grupowania uzyskuje się, stosując algorytm UPGMA.

### 2.2.2 Dzielące algorytmy hierarchiczne

Ta kategoria algorytmów hierarchicznych jest bardziej złożona od algorytmów aglomeracyjnych, gdyż wymaga zdefiniowania podprocedury tworzącej płaski podział pewnej grupy na dwie podgrupy. Problem podziału kolekcji dokumentów na  $n$  grup zostaje zatem sprowadzony do nieznacznie tylko łatwiejszego problemu podziału zbioru na 2 podgrupy. W tym celu używany jest pewien algorytm grupowania płaskiego.

Najpopularniejszym algorytmem kwalifikującym się do tej kategorii jest algorytm **dwupodziałów k-średnich** (ang. *bisecting k-means*), który do *przepełnienia* grupy wykorzystuje klasyczny algorytm k-średnich (dla  $k = 2$ ). Jako grupa do podziału (najmniej spójna) w danym kroku wybierana jest ta, w której średnia podobieństw pomiędzy dwoma różnymi należącymi do niej dokumentami ( $\frac{\sum_{d_1, d_2 \in g, d_1 \neq d_2} \text{sim}(d_1, d_2)}{|g|(|g|-1)}$ ) jest najmniejsza (istnieje również wersja algorytmu wybierająca do podziału zawsze najbardziej liczną grupę).

Artykuł [SKK00] wskazuje, że algorytm ten daje rezultaty zdecydowanie lepsze niż zwykła metoda k-średnich oraz nie gorsze od wyników algorytmów aglomeracyjnych, mając jednocześnie liniową złożoność obliczeniową względem liczby dokumentów, co w dużych kolekcjach dokumentów stanowi istotną przewagę nad algorytmami aglomeracyjnymi o złożoności kwadratowej.

### 2.3 Algorytmy grupowania oparte na gęstości

Wszystkie opisane do tej pory algorytmy znajdowały jedynie klastry będące zbiorami wypukłymi<sup>1</sup>, w szczególności metody płaskie rozpoznawały tylko klastry o kształcie sferycznym (skupienia dokumentów wokół jednego punktu). Inaczej jest w przypadku metod bazujących na gęstości w przestrzeni, które rozpoznają skupienia mające różne kształty, nawet niewypukłe. Do tej kategorii należy algorytm DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*), zaproponowany w [EpKSX96]. Do jego przedstawienia konieczne jest wprowadzenie poniższych definicji.

**Definicja 1** ( $\epsilon$ -sąsiedztwo).  $\epsilon$ -sąsiedztwem dokumentu  $d$  nazywamy zbiór dokumentów korpusu (z pominięciem dokumentu  $d$ ), które są podobne do dokumentu  $d$  w stopniu co najmniej  $(1 - \epsilon)$

$$N(d) = \{d^* \in C : \text{sim}(d, d^*) > (1 - \epsilon) \wedge d \neq d^*\}$$

**Definicja 2** (bezpośrednia osiągalność gęstościowa). Dokument  $d_2$  jest osiągalny gęstościowo z  $d_1$ , jeżeli  $d_2$  znajduje się w  $\epsilon$ -sąsiedztwie  $d_1$  oraz sąsiedztwo to zawiera co najmniej  $MinPts$  dokumentów ( $MinPts$  jest pewną stałą).

$$DDR(d_1, d_2) \Leftrightarrow d_2 \in N(d_1) \wedge |N(d_1)| > MinPts$$

**Definicja 3** (osiągalność gęstościowa). Dokument  $d_n$  jest osiągalny gęstościowo z dokumentu  $d_1$ , jeżeli istnieje ciąg dokumentów  $d_1, d_2, \dots, d_n$  taki, że każda para sąsiednich dokumentów jest bezpośrednio osiągalna gęstościowo.

$$DR(d_1, d_n) \Leftrightarrow \exists_{d_2, d_3, \dots, d_{n-1}} (RDD(d_1, d_2) \wedge RDD(d_2, d_3) \wedge \dots \wedge RDD(d_{n-1}, d_n))$$

<sup>1</sup>Dokumenty są reprezentowane przez wektory, więc pojęcie *zbiór wypukły* należy tu rozumieć zgodnie z definicją algebry liniowej, czyli:  $\forall_{d_1, d_2 \in g} \forall_{\alpha_1, \alpha_2 \geq 0, \alpha_1 + \alpha_2 = 1} (\alpha_1 \vec{d}_1 + \alpha_2 \vec{d}_2) \in g$

**Definicja 4** (połączenie gęstościowe). *Dokument  $d_2$  jest połączony gęstościowo z dokumentem  $d_1$ , jeżeli istnieje dokument  $d_3$  taki, że  $d_1$  i  $d_2$  są osiągalne gęstościowo z  $d_3$ .*

$$DC(d_1, d_2) \Leftrightarrow \exists_{d_3} (DR(d_3, d_1) \wedge DR(d_3, d_2))$$

Relacja połączenia gęstościowego jest symetryczna i przechodnia (nie jest relacją równoważności, gdyż nie jest zwrotna) a zbiór łączonych przez nią dokumentów jest traktowany przez algorytm DBSCAN jako grupa. Dokumenty, które nie należą do  $\epsilon$ -sąsiedztwa żadnego innego dokumentu i żaden dokument nie należy do ich  $\epsilon$ -sąsiedztwa, nie są połączone gęstościowo z żadnym innym dokumentem, nie należą zatem do żadnej grupy (są traktowane jako szum).

Główną wadą tego algorytmu jest trudność określenia prawidłowej wartości  $\epsilon$  dla różnych danych wejściowych. Ponadto, jeżeli podobieństwo między dokumentami w różnych grupach będzie różne (czyli klastry będą miały różną gęstość), mogą one nie być one poprawnie rozpoznane. Ma on jednak dość niską złożoność obliczeniową, wynoszącą  $O(m \log m)$ : dla każdego dokumentu ( $m$  razy) wykonywana jest procedura znajdowania jego  $\epsilon$ -sąsiedztwa, której złożoność czasowa przy odpowiednio zaindeksowanych danych wynosi  $\log m$ .

## 2.4 Algorytmy oparte na frazach

Opisane do tej pory algorytmy były klasycznymi metodami używanymi w analizie skupień, stosowanymi dla wszystkich rodzajów grupowanych danych, które można reprezentować w postaci wektorowej. Istnieją jednak również algorytmy stworzone specjalnie dla dokumentów tekstowych, które nie wymagają do działania uprzedniego sprowadzenia tekstu do postaci wektorowej, dzięki czemu możliwe jest przechowywanie informacji o kolejności wyrazów w tekstach.

### 2.4.1 Algorytm Suffix Tree Clustering

Algorytm Suffix Tree Clustering (STC) zaproponowany został w pracy [ZE05]. Istota jego działania opiera się na grupowaniu tych dokumentów, które zawierają takie same frazy (ciągi wyrazów) — one będą stanowić etykietę danego klastra. Do ich zidentyfikowania wykorzystywana jest reprezentacja dokumentów poprzez **drzewa sufiksowe** (ang. *suffix tree*). W przeciwieństwie do reprezentacji wektorowej, traktującej dokument jako multizbiór słów i pomijającej ich kolejność, drzewa sufiksowe zachowują informacje o wszystkich ciągach wyrazów, które pojawiły się w dokumentach.

**Definicja 5** (sufiks). *Sufiksem ciągu  $[a_1, a_2, \dots, a_n]$  nazywamy każdy ciąg postaci  $[a_k, a_{k+1}, \dots, a_n]$ , gdzie  $1 \leq k \leq n$ .*

---

**Algorytm 2** Algorytm DBSCAN

---

**Wejście:** zbiór dokumentów  $C = \{d_1, d_2, \dots, d_m\}$ ; minimalna wartość podobieństwa dokumentów, by zostały uznane za bliskie:  $sim_{min}$ **Wyjście:** podział  $G$ 

```

 $G \leftarrow \emptyset$ 
for  $i = 1$  to  $m$  do
  if  $visited[d_i] \neq true$  then
     $visited[d_i] \leftarrow true$ 
     $N \leftarrow \{d \in C : sim(d_i, d) > sim_{min} \wedge d_i \neq d\}$  //  $N$  jest sąsiedztwem  $d_i$ 
    if  $|N| \geq MinPts$  then
       $Cluster \leftarrow expandCluster(d, N)$ 
       $G \leftarrow G \cup Cluster$ 
    end if
  end if
end for
return  $G$ 

```

**Podprocedura** expandCluster:**Wejście:** dokument początkowy  $d_p$  i jego sąsiedztwo  $N$ **Wyjście:** grupa podobnych dokumentów

```

 $Cluster \leftarrow \{d_p\}$ 
for all  $d \in N$  do
  if  $visited[d] \neq true$  then
     $visited[d] \leftarrow true$ 
     $N' \leftarrow \{d \in C : sim(d_i, d) > sim_{min} \wedge d_i \neq d\}$  //  $N'$  jest sąsiedztwem  $d_i$ 
    if  $|N'| \geq MinPts$  then
       $N \leftarrow N \cup N'$ 
    end if
  end if
if  $d$  nie należy do żadnej grupy then
   $Cluster \leftarrow Cluster \cup \{d\}$ 
end if
end for
return  $Cluster$ 

```

---

**Definicja 6** (podciąg). *Podciągiem ciągu  $[a_1, a_2, \dots, a_n]$  nazywamy każdy ciąg postaci  $[a_k, a_{k+1}, \dots, a_j]$ , gdzie  $1 \leq k \leq j \leq n$ .*

**Definicja 7** (drzewo sufiksowe). *Drzewem sufiksowym dla ciągu  $[a_1, a_2, \dots, a_n]$  nazywamy drzewo, którego krawędzie są etykietowane podciągami tego ciągu i istnieje jednoznaczna odpowiedniość pomiędzy liśćmi drzewa a sufiksami ciągu, tzn. każdy sufiks można otrzymać przez konkatenację etykiet krawędzi na ścieżkach prowadzących od korzenia do liścia i każda taka konkatenacja jest sufiksem.*

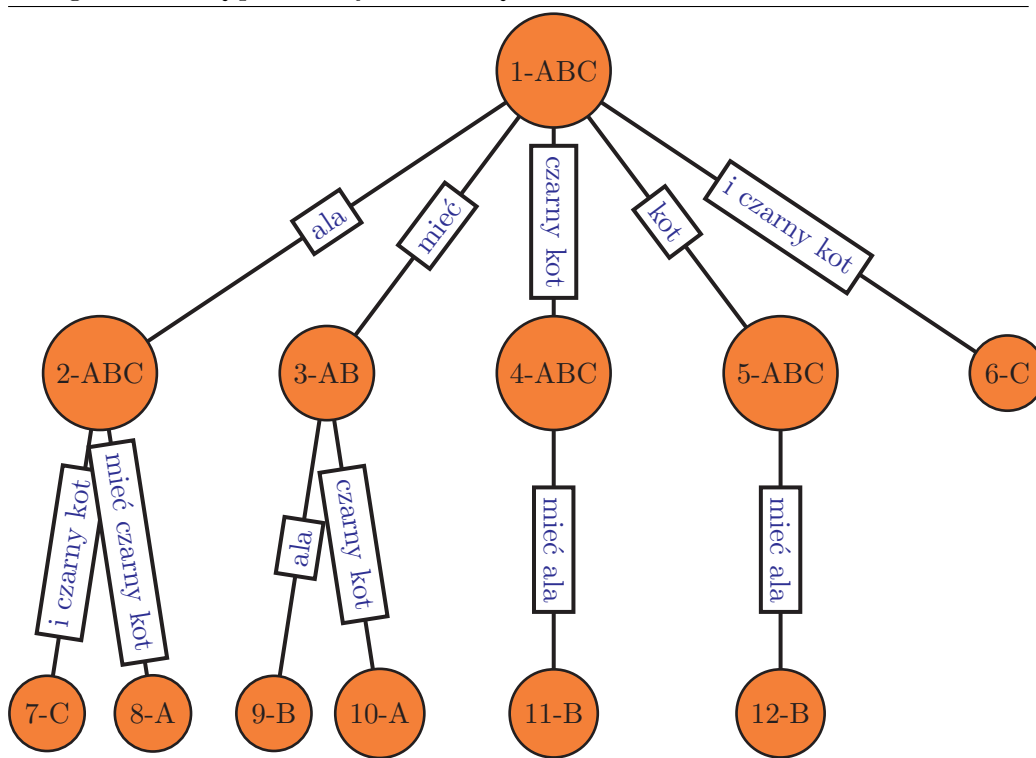
**Definicja 8** (uogólnione drzewo sufiksowe). *Uogólnione drzewo sufiksowe to drzewo sufiksowe utworzone dla zbioru ciągów zamiast pojedynczego ciągu, reprezentujące sufiksy wszystkich ciągów tego zbioru. Informacja, z którego ciągu dany sufiks pochodzi, zapisywana jest w węzłach drzewa.*

*Przykładowe uogólnione drzewo sufiksowe jest zaprezentowane na rysunku 2.*

---

**Rysunek 2** Uogólnione drzewo sufiksowe dla trzech zdań: A: *Ala ma czarnego kota.* B: *Czarny kot ma Alę.* C: *Ala i czarny kot.* Zdania zostały podzielone na wyrazy, a te sprowadzone do formy podstawowej. Informacje o pochodzeniu danego sufiksu są przechowywane w węzłach drzewa.

---



Algorytm STC składa się z trzech kroków:

1. Najpierw tworzone jest uogólnione drzewo sufiksowe dla wszystkich zdań w poszczególnych dokumentach korpusu. W węzłach zapisywana jest informacja, z którego dokumentu dany ciąg słów pochodzi.



2. W kolejnym etapie tworzona jest lista klastrów bazowych. Z powstałego w pierwszym kroku drzewa wybierane są te węzły, które opisują ciągi słów występujące w co najmniej 3 dokumentach oraz mające odpowiednio dużą ilość punktów obliczonych poniższą funkcją:

$$score(m) = |m| \cdot f(|m|) \cdot \sum_{w \in m} w_{tf-idf}(w)$$

$m$  — ciąg tokenów na ścieżce z korzenia do danego węzła

$|m|$  — długość ciągu  $m$

$f(|m|)$  — funkcja, która deprecjonuje krótkie frazy — powinna być liniowo rosnąca dla wartości o 1 do 6 i stała dla wartości większych

3. Ostatnim krokiem algorytmu jest scalanie klastrów bazowych. W tym celu budowany jest graf, którego wierzchołkami są klastry bazowe. Dwa klastry bazowe zostają połączone krawędzią, jeżeli liczba zapisanych w nich wspólnych dokumentów jest większa od wybranego wcześniej progu scalania. Spójne podgrafy utworzonego w ten sposób grafu są wynikowymi klastrami.

Algorytm STC został stworzony do grupowania wyników wyszukiwania, czyli krótkich fragmentów dokumentów (tzw. snippetów) zawierających wyszukiwane słowo. Zastosowanie go do dłuższych dokumentów i większych ich kolekcji daje wyniki niższej jakości niż algorytmy hierarchiczne i płaskie [AF96].

## 2.5 Algorytmy oparte na ukrytym indeksowaniu semantycznym

**Ukryte indeksowanie semantyczne** (ang. *Latent semantic indexing, LSI*) jest techniką ekstrakcji informacji z tekstów, która poprzez redukcję wymiarowości macierzy terminów usiłuje odtworzyć semantyczną strukturę zawartości dokumentów. Jej zastosowanie w grupowaniu dokumentów jest wciąż tematem badań [MRS08]. Oryginalna  $n$ -wymiarowa macierz terminów aproksymowana jest macierzą  $k$ -wymiarową ( $k \ll n$ ) poprzez jej **rozkład na wartości szczególne** (ang. *Singular Value Decomposition, SVD*). Pozwala to na zignorowanie terminów rzadko pojawiających się w tekstach (i przez to stanowiących jedynie szum informacyjny) oraz rozpoznanie zestawów terminów, które występują w wielu dokumentach razem i przez to prawdopodobnie są powiązane semantycznie, tworząc temat grupy dokumentów.

### 2.5.1 Algorytm Lingo

Algorytm Lingo autorstwa Stanisława Osińskiego i Dawida Weissa [Wei06] jest algorytmem stworzonym do grupowania wyników wyszukiwań (zgodnie z moją

wiedzą, nie był on na szerszą skalę badany dla pełnych tekstów dokumentów). Idea jego działania opiera się na założeniu, iż najważniejsze jest dostarczenie poprawnych (gramatycznie i logicznie) i treściwych etykiet dla klastrów. Osoba oglądająca pogrupowane wyniki powinna móc przede wszystkim szybko zapoznać się z przekrojem tematycznym znalezionych dokumentów. Dużą wagę przykładają się do poprawności gramatycznej i zrozumiałości etykiet, gdyż — jak pokazały badania przeprowadzone przez pomysłodawców tego podejścia — czytelnik szybko się zniechęca, gdy ich zrozumienie wymaga wysiłku. Dopuszczalne jest przypisanie dokumentu do kilku grup (gdyż dokument często dotyczy kilku różnych tematów) lub nieprzypisanie do żadnej. Struktura klastrów jest płaska, nie hierarchiczna, gdyż informacja o powiązaniach między grupami uznawana jest za zbędną.

Metodą zastosowaną w celu spełnienia wyżej opisanych warunków jest technika **Description Comes First**, zaprezentowana w [Wei06]. Polega ona na odwróceniu tradycyjnej kolejności etapów grupowania: najpierw tworzone są etykiety dla klastrów, a następnie do nich przypisywane są dokumenty. Unika się w ten sposób trudnego zadania polegającego na tworzeniu deskryptywnych opisów dla grup dokumentów. Algorytm Lingo realizuje ten schemat w następujących etapach:

**wstępne przetworzenie tekstu** — ze stokenizowanego tekstu dokumentu wejściowego usunięte zostają znaki nieliterowe (poza znakami końca zdania) oraz słowa funkcyjne, pozostałe wyrazy zostają sprowadzone do formy podstawowej

**znajdowanie częstych fraz** — za pomocą tablic sufiksowych<sup>2</sup> wyszukiwane są frazy spełniające następujące warunki:

- pojawiają się w odpowiednio dużej liczbie dokumentów
- pochodzą z jednego zdania (nie są brane pod uwagę ciągi słów, które stanowią koniec jednego zdania i początek następnego).
- składają się z możliwie wielu wyrazów
- nie zaczynają ani nie kończą się słowem funkcyjnym (takowe mogą się pojawić w środku frazy)

**tworzenie etykiet dla grup** — najpierw wykrywane są dominujące tematy (reprezentowane zbiorem terminów) opisujące wejściową kolekcję dokumentów. Uzyskuje się je przy pomocy rozkładu macierzy terminów na wartości szczególne. Następnie do tych tematów przypisywane są uzyskane w poprzednim

---

<sup>2</sup>Tablice sufiksowe są strukturą danych przechowującą wszystkie sufiksy rozpatrywanego ciągu — pozwalają reprezentować te same informacje co drzewa sufiksowe, lecz wykorzystują mniej pamięci

kroku częste frazy (poprzez obliczenie i porównanie podobieństwa  $sim_{cosine}$ ) i frazy te stają się zrozumiałymi dla człowieka etykietami dla grup

**przypisywanie dokumentów do grup** — dokumenty i etykiety otrzymane w poprzednim kroku sprowadzane są do reprezentacji wektorowej. Jeżeli podobieństwo (np.  $sim_{cosine}$ ) pomiędzy pewnym dokumentem i etykietą osiągnie wystarczająco dużą wartość, to dokument ten jest przypisywany do klastra z tą etykietą

**sortowanie grup** — ostatnią fazą jest sortowanie grup w taki sposób, by te najliczniejsze i posiadające najbardziej deskryptywne opisy występowały na początku.

## Rozdział 3

---

# Metody doskonalenia reprezentacji dokumentów

---

Klasyczne algorytmy grupowania traktują dokumenty tekstowe jako zwykłe multizbiory słów, nie analizując lingwistycznie przetwarzanego tekstu. Zaletą tego rozwiązania jest oczywiście łatwość jego implementacji, uniwersalność (ta sama aplikacja będzie działać dla wielu różnych języków) oraz szybkość działania. Wadą jest niższa (zwykle niestety niedostateczna) jakość otrzymanych wyników.

Aby poprawić wektorowe reprezentacje dokumentów stosowane są różne metody bazujące na własnościach języka naturalnego. Umożliwiają one ekstrakcję informacji charakterystycznych dla danego dokumentu, przyczyniając się do generowania trafniejszych wyników grupowania.

### 3.1 Pomijanie wyrazów funkcyjnych

Języki naturalne obfitują w wyrazy o charakterze pomocniczym, funkcyjnym. Są to najczęściej spójniki typu: *i*, *oraz*, *że*. Nie niosą one same w sobie informacji o tematyce tekstu, nie są zatem przydatne w procesie grupowania dokumentów — stanowią szum zaburzający poprawne wyniki. Nazywamy je **wyrazami funkcyjnymi** (ang. *stop words*).

Pomijanie wyrazów funkcyjnych jest bardzo prostą metodą poprawiania jakości reprezentacji tekstów, która stosowana jest od dawna w niemal wszystkich aplikacjach z dziedziny pozyskiwania informacji. Najczęściej wszystkie wyrazy funkcyjne dla konkretnego języka zbierane są na specjalnej liście, dzięki czemu późniejsze ich rozpoznanie ogranicza się do odszukania na tej liście.

Zbiory słów funkcyjnych mogą być tworzone przez ludzi w sposób ręczny, na podstawie ich intuicji czy doświadczenia. Można je również otrzymać w spo-

sób automatyczny, poprzez wybranie początkowego fragmentu listy frekwencyjnej stworzonej na bazie dostatecznie dużego korpusu.

## 3.2 Lematyzacja

Istotnym etapem przetwarzania tekstu jest **lematyzacja**, czyli sprowadzenie różnych form morfologicznych pewnego leksemu do jego formy kanonicznej (inaczej zwanej lematem lub lemmą). Różne formy fleksyjne wyrazów są konieczne do zbudowania poprawnego syntaktycznie zdania i wskazują na odpowiednią funkcję danego wyrazu w zdaniu, jednak z punktu widzenia poprawnej reprezentacji dokumentu, słowo *piłkę* oraz *piłce* niosą tę samą informację, więc powinny być rozpoznane jako wystąpienie tego samego leksemu. Proces lematyzacji jest szczególnie ważny dla języków o bogatej fleksji, do których się zalicza język polski. Lematyzatory najczęściej opierają się na słowniku zawierającym pary postaci: forma morfologiczna — leksem.

Do reprezentacji tekstu tworzonej wyłącznie w celu obliczenia podobieństwa pomiędzy dwoma dokumentami nie jest konieczne znalezienie poprawnej formy kanonicznej wyrazu, wystarczy jego **rdzeń** (ang. *stem*), czyli taka jego częśćka, która umożliwi jednoznaczne zidentyfikowanie leksemu. Proces sprowadzenia wyrazu do jego rdzenia nazywamy **stemmingiem**. Klasycznym przykładem tej procedury jest algorytm Portera [Por80], który znajduje rdzenie dla słów angielskich. Tego typu algorytmy tworzone są zwykle w oparciu o reguły specjalnie skonstruowane dla konkretnego języka.

Zauważyć należy, że efektem działania lematyzatora nie może być jedynie forma podstawowa wejściowego wyrazu, gdyż istnieją pary różnych leksemów o jednakowym lemacie. Informacja o rozpoznanym leksemie powinna zawierać, oprócz lematu, również jego część mowy (która może być podstawą dalszego poprawienia reprezentacji dokumentu, co zostało opisane w rozdziale 3.3) i ewentualnie numer kolejnego homonimu (jeżeli istnieje kilka leksemów o jednakowej formie podstawowej i części mowy, np. *zamek*–budowla warowna i *zamek*–suwak).

## 3.3 Selekcja wyrazów o określonej części mowy

W rozdziale 3.1 pokazano, że nie wszystkie wyrazy występujące w tekście powinny się pojawić w wektorowej reprezentacji dokumentu, gdyż część z nich nie niesie istotnych informacji. Rozszerzeniem opisanej tam metody pomijania wyrazów funkcyjnych jest selekcja terminów będących określoną częścią mowy.

Z punktu widzenia grupowania dokumentów, największe znaczenie mają rzeczowniki - to one nazywają byty i wskazują na tematykę tekstu. Istotne są również przymiotniki jako wyrazy opisujące cechy i właściwości rzeczowników. Niełatwo jest ocenić przydatność czasowników. Wiele z nich ma charakter jedynie funkcyj-

ny: poza oczywistym przykładem wyrazu *być* można tu wskazać czasowniki *lubić*, *zajmować się* czy *uważać*. Z drugiej strony, istnieją czasowniki, które wyraźnie wskazują na tematykę dokumentu: leksem *różniczkować* jest charakterystyczny dla tekstów z dziedziny analizy matematycznej, zaś *debugować* dla tekstów związanych z programowaniem. Prawdopodobnie jednak większość czasowników ma znaczenie zbyt ogólne, by na nich opierać funkcję podobieństwa. Jeszcze mniejszą wagę należy przypisać przysłówkom, które najczęściej opisują czasowniki. Leksemy należące do pozostałych części mowy (spójniki, zaimki, przymyki) nie niosą z reguły żadnych informacji, należy je omijać podczas indeksowania dokumentu.

### 3.4 Rozpoznawanie fraz

Występujące w tekście frazy, czyli związki wyrazów stanowiących całość znaczeniową<sup>1</sup>, powinny być pojedynczymi terminami w reprezentacji dokumentów, gdyż mają one często inne znaczenie niż leksemy wchodzące w ich skład. Frazy powtarzają się w różnych dokumentach rzadziej niż pojedyncze wyrazy, dlatego są one cechami o bardzo silnych właściwościach dystynktywnych. Współwystępowanie terminów wielowyrazowych jest dowodem bliskości tematycznej tekstów i na takim założeniu opiera się algorytm STC (rozdział 2.4.1).

Rozpoznanie fraz w powyższym znaczeniu wymaga analizy składniowej zdań w grupowanych dokumentach, do czego potrzebna jest znajomość reguł gramatycznych obowiązujących w danym języku. Znajdowanie fraz może się jednak opierać na metodach statystycznych, które są stosunkowo proste do zaimplementowania, gdyż nie wymagają analizy językowej. Za frazę w rozumieniu statystycznym uważać będziemy ciąg leksemów, który pojawia się często w różnych dokumentach (nie zawsze zatem będzie on stanowić całość znaczeniową). Do wyszukania fraz w tym sensie można wykorzystać drzewa lub tablice sufiksowe (używane odpowiednio przez algorytmy STC i Lingo). Jeszcze prostszą metodą na uwzględnienie kolejności słów jest reprezentowanie w modelu wektorowym  $n$ -gramów (najczęściej bigramów lub trigramów), czyli wszystkich sekwencji  $n$  wyrazów bezpośrednio po sobie występujących w pewnym zdaniu (często pomija się wówczas wyrazy funkcyjne). Tak otrzymane ciągi słów mogą zastąpić lub uzupełnić standardową reprezentację wektorową leksemów.

#### 3.4.1 Rozpoznawanie jednostek nazwanych

**Jednostką nazwaną** (ang. *named entity*) nazywamy sekwencję wyrazów, które odnoszą się do pojedynczych bytów takich jak osoby, organizacje czy lokalizacje geograficzne [GJM09]. Ich prawidłowa reprezentacja w wektorze terminów jest wyjątkowo istotna, gdyż poprawnie rozpoznane stanowią bardzo wartościowe

<sup>1</sup>Definicja za Słownikiem Języka Polskiego, <http://www.sjp.pwn.pl> (maj 2010)

cechy (rzadko występują i mają bardzo wąskie znaczenie), zaś traktowanie wyrazów wchodzących w ich skład jako osobne terminy może spowodować zmianę ich znaczenia i konsekwentnie pogorszenie reprezentacji — dla przykładu nazwy miejscowości: *Zimna Wódka* czy *Nowe Rumunki* niosą zupełnie inną treść niż wyrazy wchodzące w ich skład.

Jednostki nazwane mogą odnosić się również do innych typów informacji, takich jak ilości (np. *15 kilogramów*) czy daty (np. *14. kwietnia 2010 roku*), które również mogą być przydatne w reprezentacji dokumentów, lecz powinny zostać najpierw znormalizowane, gdyż np. daty mogą być wyrażane na wiele różnych sposobów, co utrudnia ich porównywanie (dla przykładu: terminy *10 kwietnia 2010* i *10 IV 2010* nie będą rozpoznane jako ta sama cecha, mimo że wskazują one na tę samą okoliczność).

### 3.5 Semantyka

Uwzględnienie informacji semantycznej w reprezentacji dokumentów tekstowych może w znaczny sposób poprawić jakość procesu grupowania. Intuicja podpowiada, że dokument, w którym często pojawia się słowo *piłkarz*, ma podobną tematykę do dokumentu z powtarzającym się wyrazem *futbolista* — oba opowiadają o osobie grającej w piłkę nożną. Zwykły model multizbioru słów nie jest w stanie tego podobieństwa oddać i wskazałby, że wspomniane dokumenty nie posiadają cech wspólnych. Opiera się on bowiem na uproszczeniu zakładającym wzajemną jednoznaczność pomiędzy zbiorem leksemów i **konceptów** (pojęć, czyli znaczeń leksemów). Wiadomo jednak, że jeden leksem może opisywać różne koncepty (zjawisko polisemii), jeden koncept może również być reprezentowany przez różne leksemy (synonimia).

Wykorzystanie informacji semantycznej jest możliwe dzięki reprezentacji wiedzy o otaczającym nas świecie — **komputerowej ontologii**. Ontologia opisuje koncepty i relacje pomiędzy nimi. Najpopularniejszą ontologią jest stworzony pierwotnie dla języka angielskiego WordNet (powstały również wzorujące się na WordNecie ontologie dla innych języków, w tym dla języka polskiego). Koncepty reprezentowane są w nim jako **synsety**, czyli zbiory leksemów o bardzo zbliżonym znaczeniu (synonimów). Synsety są powiązane między sobą różnymi relacjami. Z perspektywy reprezentacji dokumentów, najważniejsze są relacje łączące synsety zawierające rzeczowniki<sup>2</sup>, czyli:

**hiperonimia**  $A$  jest hiperonimem dla  $B$ , jeżeli  $A$  jest synsetem reprezentującym pojęcie nadrzędne w stosunku do  $B$  (np. *zwierzę* jest hiperonimem dla *ssak*)

**hiponimia**  $A$  jest hiponimem dla  $B$ , jeżeli  $B$  jest hiperonimem dla  $A$

<sup>2</sup>WordNet posiada synsety zawierające rzeczowniki, czasowniki, przymiotniki i przysłówki, dla każdego z tych typów synsetów zdefiniowane są inne rodzaje relacji.

**meronimia**  $A$  jest meronimem dla  $B$ , jeżeli  $A$  jest synsetem reprezentującym pojęcie będące częścią  $B$  (np. *okładka* jest meronimem dla *książka*)

**holonimia**  $A$  jest holonimem dla  $B$ , jeżeli  $B$  jest meronimem dla  $A$

**antonimia**  $A$  jest antonimem dla  $B$ , jeżeli  $A$  i  $B$  mają znaczenia przeciwstawne (relacja symetryczna)

Synsety, które mają taki sam hiperonim, nazywane są **siostrzanymi**. Do siostrzanych synsetów należą na przykład wyrazy: *kanarek*, *papuga*, *gołąb* i *wróbek*, gdyż wszystkie połączone są relacją hiperonimii z synsetem *ptak*.

### 3.5.1 Ujednoznacznianie leksemów

Zidentyfikowanie konceptów występujących w dokumencie jest zagadnieniem trudnym, gdyż duża część leksemów rzeczownikowych jest wieloznaczna i należy do kilku synsetów<sup>3</sup>. Automatyczne rozpoznanie właściwego znaczenia jest jednak bardzo trudne. Czasami umożliwia to analiza składniowa zdania — czasowniki często łączą się z rzeczownikiem w konkretnym znaczeniu.

Jednym ze sposobów na znalezienie właściwych znaczeń dla wyrazów występujących w tekście jest metoda bazująca na **słowach wspierających** [For06]. Dla każdego synsetu przypisanego do wieloznacznego leksemu tworzy się zbiór wyrazów powiązanych z danym synsetem, tzn. połączonych z nim jakąś relacją, czyli hiperonimów, hiponimów, antonimów, meronimów itd. Następnie zliczane są wystąpienia tych słów w całym dokumencie i wybierany jest ten synset, dla którego tych wystąpień było najwięcej.

Metodą prostszą od wyżej opisanej jest stworzenie stałego mapowania leksemu na synset na podstawie częstości wykorzystania wyrazu w danym znaczeniu. Wyraz *powód* w ogólnej kolekcji dokumentów częściej będzie oznaczał *przyczynę* niż *osobę wnoszącą sprawę do sądu czy rzemień, na którym prowadzi się konia*<sup>4</sup>, dlatego najmniej błędów popełnimy zaliczając ów wyraz zawsze do pierwszego synsetu.

### 3.5.2 Uwzględnienie relacji semantycznych w reprezentacji wektorowej

Analogicznie do wektora terminów  $\vec{d}$  reprezentującego dokument  $d$  możemy zdefiniować wektor synsetów  $\vec{d}^s$  dla tegoż dokumentu:

$$\vec{d}^s = [w(s_1), w(s_2), \dots, w(s_r)]$$

<sup>3</sup>W polskim WordNecie (WNPl) około 24% rzeczowników jest wieloznaczna, tzn. należy do kilku synsetów

<sup>4</sup>Definicje za Słownikiem Języka Polskiego, <http://www.sjp.pwn.pl> (maj 2010)



gdzie  $\{s_1, s_2, \dots, s_r\}$  jest zbiorem synsetów zidentyfikowanych w danym dokumencie oraz  $w(s)$  jest funkcją ważącą dla synsetów, oznaczającą sumę wag terminów należących do danego synsetu i występujących w dokumencie:

$$w(s) = \sum_{t \in s} w(t) \quad (3.1)$$

Waga  $w(s)$  może być również rozszerzona w taki sposób, by uwzględniała relacje zachodzące między synsetami. Największe korzyści może przynieść uwzględnienie relacji hiponimii w ten sposób, by wagę synsetu powiększyć o wagę wszystkich jego hiponimów:

$$w(s) = \sum_{t \in s} w(t) + \sum_{s^* \in S: s^* \prec s} \sum_{t \in s^*} w(t) \quad (3.2)$$

gdzie  $S$  — zbiór wszystkich synsetów,  $s_1 \prec s_2 \Leftrightarrow s_1$  jest hiponimem dla  $s_2 \vee \exists s_3 s_1 \prec s_3 \wedge s_3 \prec s_2$ .

Tak utworzony wektor synsetów może posłużyć do polepszenia reprezentacji wektorowej na kilka sposobów (za [HSS03]):

**dodanie synsetów** — konkatencja wektora terminów i wektora synsetów dla dokumentu  $d$

**zastąpienie terminów synsetami** — podobnie jak w przypadku dodania synsetów, lecz usuwane są wagi terminów, które występują w którymś z rozpoznanych synsetów

**sam wektor synsetów** — reprezentacją dokumentu jest wektor synsetów.

## Rozdział 4

---

# Implementacja metod grupowania dokumentów

---

Niniejszy rozdział opisuje implementację programu grupującego dokumenty tekstowe, wykorzystującego opisane w poprzednich rozdziałach algorytmy oraz metody reprezentacji.

Aplikacja napisana została w języku Java SE 6, za warstwę prezentacji odpowiada biblioteka Swing z rozszerzeniem SwingX oraz Substance. Przeznaczeniem aplikacji jest prezentacja działania algorytmów oraz sposobów reprezentacji dokumentów w celach badawczo-naukowych. Została jej nadana nazwa **Ugrupator**.

### 4.1 Reprezentacja dokumentów

Ponieważ niniejsza praca stawia sobie za cel zaproponowanie metod ulepszających reprezentację dokumentów, program oferuje możliwość wektorowego modelowania wejściowych tekstów przy pomocy powszechnie stosowanych płytkich metod oraz metod bardziej zaawansowanych, wykorzystujących mechanizmy przetwarzania języka naturalnego oraz metody ekstrakcji informacji. W pierwszym przypadku wektor terminów powstaje przez tokenizację, lematyzację i usunięcie słów funkcyjnych z tekstu, druga zaś dodatkowo analizuje zidentyfikowane leksemy pod kątem ich części mowy, rozpoznaje frazy i nazwy własne, a także modeluje semantykę za pomocą ontologii typu WordNet.

#### 4.1.1 Standardowa reprezentacja dokumentu

Pierwszym etapem tworzenia standardowej reprezentacji dokumentu jest tokenizacja zawartego w nim tekstu. Podział na tokeny realizowany jest za pomocą prostej zasady wyszukiwania ciągów składających się wyłącznie z liter (w szcze-

gólności oznacza to, iż wyrazy złożone pisane z dywizem traktowane są jako odrębne tokeny), zrealizowany za pomocą wyrażenia regularnego w standardzie Unicode, użytego w następującej metodzie Javy:

```
public static final String [] tokenize(String input) {
    return input.split("\\P{L}+");
}
```

Ponieważ język polski cechuje się bogatą fleksją, przeprowadzenie lematyzacji jest konieczne, by wektory terminów poprawnie reprezentowały treść modelowanych dokumentów. Do tego celu użyty został lematyzator<sup>1</sup> będący częścią biblioteki morfologik<sup>2</sup>. Oparty jest on na automacie skończonym z wyjściem, znajdującym formy podstawowe i części mowy dla wejściowych wyrazów. Podczas przetwarzania dokumentu, dla każdego wyrazu jest wyszukiwana jego forma podstawowa i, jeżeli zostanie ona znaleziona, to ona reprezentowana jest w wektorze terminów. Wystąpienia homografów<sup>3</sup>, dla których proponowanych jest kilka leksemów, traktowane są jak wystąpienie wszystkich możliwych ich interpretacji. Oznacza to, że jeżeli w tekście pojawi się słowo *trybuna*, to do wektora terminów dodane zostaną dwa leksemy: *trybun* oraz *trybuna*. Wyjątek stanowią homografy leksemów o jednakowej formie kanonicznej (np. *niż*–spójnik oraz *niż*–rzeczownik) — zliczane są one jako jeden leksem, gdyż są one nierozróżnialne w reprezentacji przechowującej jedynie formę podstawową leksemów.

Z otrzymanego w powyżej opisany sposób multizbioru leksemów odrzucane są te, które występują na liście słów funkcyjnych. Lista ta została stworzona w oparciu o stop-listę polskiej Wikipedii<sup>4</sup> oraz zbiór 200 leksemów o największej frekwencji w części demonstracyjnej Korpusu Języka Polskiego Wydawnictwa Naukowego PWN<sup>5</sup>. Na tej liście frekwencyjnej znajdują się nie tylko wyrazy funkcyjne w tradycyjnym rozumieniu — od 41. miejsca pojawiają się na niej popularne rzeczowniki takie jak *czas* czy *człowiek*, które wprawdzie niosą pewną informację, lecz ze względu na swoją wysoką frekwencję i szerokie zastosowanie nie stanowią dobrych cech dla porównywania dokumentów, dlatego również nie są uwzględniane w wektorze terminów.

Formy podstawowe pozostałych leksemów oraz tokeny, których nie udało się zlematyzować, tworzą zbiór terminów dla danego dokumentu.

<sup>1</sup>Przez autora, Dawida Weissa, nazwany przewrotnie lematyzatorem (od angielskiego *lame* — kiepski)

<sup>2</sup><http://morfologik.blogspot.com/>, dostęp maj 2010

<sup>3</sup>Homografy to wyrazy o jednakowej pisowni, które są formą różnych leksemów

<sup>4</sup><http://pl.wikipedia.org/wiki/Wikipedia:Stopwords>, dostęp czerwiec 2010

<sup>5</sup><http://korpus.pwn.pl/stslow.php>, dostęp czerwiec 2010

### 4.1.2 Reprezentacja dokumentu oparta na metodach analizy językowej

Rozszerzona reprezentacja dokumentów opiera się na wyniku działania parsera systemu tłumaczenia automatycznego Translatica. Trwa ona stosunkowo długo, dlatego nie została zintegrowana z aplikacją grupującą — musi być dokonana przed rozpoczęciem pracy z programem, a jej wynik zapisany w postaci plików w formacie ITF<sup>6</sup>. Struktura ta przechowuje wiele danych o przeanalizowanych zdaniach. W celu stworzenia reprezentacji dokumentu wykorzystywana jest część z nich: zidentyfikowane leksemy oraz frazy wraz z przypisanymi do nich formami podstawowymi i częściami mowy. To one stanowią początkowo zbiór terminów dla danego dokumentu. Usunięte z niego zostają te, które znajdują się na liście wyrazów funkcyjnych (tej samej co w reprezentacji standardowej). Zaznaczyć należy, że leksemy wchodzące w skład fraz są również traktowane jako osobne terminy — w przeciwnym przypadku wystąpienie frazy *film fabularny* i leksemu *film* nie byłoby rozpoznane jako cecha podobieństwa.

#### System ważenia terminów

Do zbioru terminów reprezentujących dany dokument trafiają jedynie jednostki o określonej części mowy: rzeczowniki, przymiotniki, czasowniki i przysłówki. Jak jednak zostało pokazane w rozdziale 3.3, nie każda z nich niesie równie dużo informacji. By móc ten fakt uwzględnić w reprezentacji, wykorzystany został tutaj system ważenia terminów. Domyślnie wagą każdego terminu jest liczba jego wystąpień  $w_{tf}$ . Dla każdej części mowy użytkownik może jednak zdefiniować **współczynnik ważenia**, czyli liczbę rzeczywistą, przez którą zostanie przemnożona waga każdego należącego do danej części mowy leksemu. W szczególności, współczynnik ważenia może być zerem, by dana część mowy nie była uwzględniana w reprezentacji. Analogiczny współczynnik można zdefiniować dla fraz, które powtarzają się w dokumentach rzadziej i przez to zwykle niosą więcej informacji niż pojedynczy leksem, zatem powinny być reprezentowane silniej niż zwykle leksemy. Współczynniki dla fraz oraz części mowy są brane pod uwagę równolegle, tzn. np. waga frazy rzeczownikowej jest wynikiem mnożenia liczby jej wystąpień przez współczynnik dla rzeczowników oraz fraz.

Szczególne znaczenie wśród terminów zarówno jedno- jak i wielowyrazowych mają nazwy własne. Rozpoznawane są one prostą heurystyką: muszą zaczynać się dużą literą i być rzeczownikiem<sup>7</sup>. Jeżeli dany termin zostanie zakwalifikowany jako nazwa własna, jego waga zostanie przemnożona tylko przez współczynnik ważenia dla nazw własnych (nie są brane pod uwagę współczynniki dla rzeczowników ani fraz).

<sup>6</sup>Internal Translatica Format, wewnętrzny format wymiany danych systemu Translatica

<sup>7</sup>Jeżeli część mowy nie zostanie rozpoznana, a rozpatrywany termin zaczyna się wielką literą, to również jest traktowany jako nazwa własna

**Rysunek 3** Ekran ustawiania współczynników ważenia w Ugrupatorze

### Wykorzystanie WordNetu

W celu analizy relacji semantycznych pomiędzy terminami rozpoznanymi w dokumencie wykorzystywana jest ontologia plWordNet, opracowywana przez zespół z Politechniki Wrocławskiej koordynowany przez Macieja Piaseckiego. Niestety, metody opisane w rozdziale 3.5 nie przyniosły oczekiwanych rezultatów — problem stanowiło przede wszystkim poprawne przypisanie polisemicznych słów do odpowiednich synsetów, zawiodły zarówno próby stałego mapowania lematu na synset, jak i prób dezambiguacji opartych na słowach wspierających. Poprawę tej sytuacji udało się uzyskać dopiero po wprowadzeniu dość rozbudowanych warunków, które muszą być spełnione, by synset został rozpoznany.

Proces uwzględniania relacji semantycznych w reprezentacji przeniesiony został z etapu indeksowania dokumentów do momentu obliczania podobieństwa pomiędzy nimi. Zanim wektory reprezentujące dokumenty zostaną porównane zgodnie z metodą opisaną w 4.1.3, niektóre terminy z tego wektora zostaną zastąpione synsetami zgodnie z następującym schematem:

1. Wyszukaj synsety, które zawierają leksemy z pierwszego i drugiego porównywanego dokumentu
2. Akceptuj tylko te synsety, które zawierają termin występujący w obu dokumentach lub wszystkie należące do nich terminy rozpoznane w obu dokumentach są monosemiczne (tzn. nie należą do żadnego innego synsetu)
3. Dla obu dokumentów oblicz wagi pozostałych synsetów poprzez zsumowanie wag terminów należących do tych synsetów oraz ich hiponimów —

zastosowano tu wzór 3.2 zmodyfikowany w ten sposób, że brane są jedynie bezpośrednie hiponimy, gdyż synsety pośrednio powiązane tą relacją często mają zbyt odległe znaczenie, by je uwzględniać

4. Dodaj synsety do wektorów odpowiednich dokumentów, usuwając z nich jednocześnie wszystkie terminy, które należą do tych synsetów

Ten sposób uwzględniania relacji semantycznych umożliwia identyfikację grup leksemów o zbliżonym znaczeniu z obu dokumentów z niewielkim ryzykiem popełnienia błędu. Umożliwia to znalezienie dodatkowych cech podobieństw zbliżonych dokumentów, nie powodując dodawania nieistniejących powiązań między różnymi się wyrazami. Powyżej opisane zmiany w reprezentacji obowiązują jedynie przy obliczaniu podobieństwa pomiędzy konkretną parą dokumentów i po tym nie są zapamiętywane.

### 4.1.3 Reprezentacja wektora terminów

Aby móc przedstawić dokument w postaci wektora w dokładnie taki sposób, jak zostało to opisane w rozdziale 1.1, konieczna jest znajomość wszystkich terminów, które pojawiają się w całej grupowanej kolekcji dokumentów. Oznacza to, że przed rozpoczęciem indeksowania dokumentów należy zebrać wszystkie terminy, które mogą się pojawić we wszystkich tekstach. Ponadto, wektory reprezentujące dokumenty są w takiej reprezentacji rzadkie<sup>8</sup>, co sprawia, że taki model wykorzystuje pamięć w sposób nieefektywny. Wady te można zniwelować, przedstawiając dokument w postaci dwóch wektorów: jeden stanowi tablicę terminów rozpoznanych w danym dokumencie, drugi zaś przechowuje informację ich o frekwencji — takie rozwiązanie zostało zaimplementowane w bibliotece Lucene<sup>9</sup> (interfejs `TermFreqVector`). Wówczas jednak odczytanie frekwencji losowego terminu wymaga wcześniejszego odnalezienia go w wektorze, co oznacza nawet  $n$  ( $n$  — długość wektora) porównań. Z tego powodu uznałem, że najwygodniejszą reprezentacją zawartości dokumentu dla potrzeb grupowania będzie tablica asocjacyjna, przechowująca pary postaci: *termin* → *waga terminu*:

```
public class Document {

    /* DefaultValueHashMap to rozszerzenie klasy HashMap,
       zwracające domyślną wartość dla zapytań o klucze
       nieistniejące w kolekcji */
    private final Map<String, Double> weightedTerms =
        new DefaultValueHashMap<String, Double>(new Double(0))
        ;
}
```

<sup>8</sup>Wektor rzadki to taki wektor, w którym większość elementów jest zerowa.

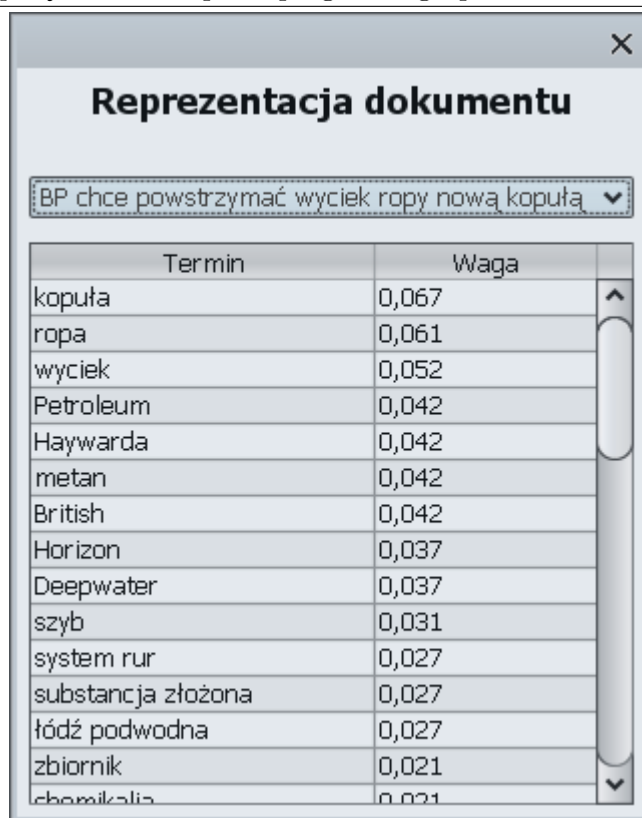
<sup>9</sup><http://lucene.apache.org/>, dostęp maj 2010

```
(...)  
}
```

Wagi terminów zostają obliczone zgodnie z zasadami opisanymi w rozdziale 4.1.2. Po skończeniu analizowania całego korpusu może nastąpić (zależy to od konfiguracji programu) korekta wartości tych wag o odwrotną częstotliwość dokumentową terminów, stanowiąc wagę  $w_{tf-idf}$ . Przykładowa reprezentacja dokumentu zaprezentowana jest na rysunku 4.

Podobieństwo pomiędzy dokumentami obliczane jest zmodyfikowaną funkcją  $sim_{cosine}$ . Owa modyfikacja opiera się na założeniu, że większe znaczenie mają te terminy, które występują w obu dokumentach (różnice w zastosowanym słownictwie mają mniejsze znaczenie) dlatego podczas obliczania wartości podobieństwa funkcją  $sim_{cosine}$  wagi tych terminów zostają zwiększone dwukrotnie (tylko na czas obliczania podobieństwa).

**Rysunek 4** Przykładowa reprezentacja dokumentu *BP chce powstrzymać wyciek ropy nową kopułą* stworzona przez program Ugrupator



Termin	Waga
kopuła	0,067
ropa	0,061
wyciek	0,052
Petroleum	0,042
Haywarda	0,042
metan	0,042
British	0,042
Horizon	0,037
Deepwater	0,037
szyb	0,031
system rur	0,027
substancja złożona	0,027
łódź podwodna	0,027
zbiornik	0,021
chemikalia	0,021

## 4.2 Zaimplementowane algorytmy grupowania

W programie zaimplementowane zostały algorytmy analizy skupień należące do różnych kategorii: płaski, hierarchiczny i oparty na gęstości. Są to odpowiednio: algorytm k-średnich, DBSCAN oraz hierarchiczny aglomeracyjny z podobieństwem grup *SIM<sub>group-average</sub>* (UPGMA). Zostały one zaimplementowane zgodnie z ich schematami podanymi w rozdziale 2.

Wstępne testy działania wymienionych algorytmów wykazały, że metoda k-średnich nie spisuje się dobrze w grupowaniu małych kolekcji dokumentów tekstowych (takich, jakie były badane) — wyniki zwracane przez nią są silnie niedeterministyczne, a powstałe grupy tylko w części zawierają dokumenty powiązane ze sobą w widoczny sposób.

DBSCAN zwracał ciekawe wyniki — przede wszystkim dlatego, że dokumenty niebędące wystarczająco podobne do innych i przez to niepasujące do żadnej grupy były odrzucane i traktowane jako szum. Dzięki temu może on być stosowany do zupełnie losowych kolekcji w celu odszukania tekstów na ten sam temat. Cechą nie zawsze pożądaną w jego działaniu jest tzw. *efekt lancuchowy* — jeżeli dwie różne grupy posiadają jedną parę podobnych dokumentów, zostaną one połączone w jedną grupę.

Algorytm aglomeracyjny UPGMA najlepiej organizuje dane, jednak format jego wyniku (drzewo) nie jest czytelny — użytkownik zwykle chce zobaczyć grupy dokumentów na jeden temat, bez rozpatrywania podobieństw między grupami i wewnątrz nich. Dlatego zaimplementowane zostały dodatkowo dwa warianty tego algorytmu, zwracające podział w postaci niehierarchicznej. Pierwszy z nich (w dalszej części pracy będzie on nazywany **HAC-Flat**) wymaga podania jako parametr liczby grup, na jakie ma być podzielona wejściowa kolekcja, i w momencie uzyskania w procesie scalania pożądanej liczby grup — kończy działanie. Drugi wariant (dalej zwany **HAC-Threshold**) przerywa scalanie klastrów wówczas, gdy podobieństwo łączonych grup spadnie poniżej określonego jako parametr progę, a pozostałe jednoelementowe grupy łączy w jedną grupę o etykiecie *Inne*. Schematy obu tych wariantów algorytmu aglomeracyjnego przedstawione zostały na listingach 4.1 i 4.2 w postaci kodów źródłowych z zaimplementowanego programu.

Listing 4.1: Algorytm grupowania aglomeracyjnego — wariant HAC-Flat

```
public class HACAlgorithmFlatResult implements ClusteringAlgorithm {
// (Tu występuje deklaracja pól i ~konstruktorów klasy, usunięta dla
// przejrzystości)
@Override
public Collection<Cluster> cluster(Map<String, Object> params)
throws WrongParamException {
// Pobranie parametrów grupowania - liczby klastrów
Integer desiredClusterCount = (Integer)
params.get(CLUSTER_COUNT.PARAM);
```



```

if (desiredClusterCount == null || desiredClusterCount >
    corpus.getDocuments().size() || desiredClusterCount < 1) {
    throw new WrongParamException("Cluster count must be between 1
        and document count!");
}

// Utwórz początkowe jednodokumentowe klastry
List<Cluster> clusters = new
    ArrayList<Cluster>(corpus.getDocuments().size());
for (Document d : corpus.getDocuments()) {
    clusters.add(new Cluster(new
        ArrayList<Document>(Arrays.asList(d))));
}

// Łącz klastry aż uzyskana zostanie pożądana ich liczba
while (clusters.size() > desiredClusterCount) {
    double clustersToJoinSimilarity = -1d;
    int firstClusterToJoinNumber = 0, secondClusterToJoinNumber = 0;
    // Znajdź parę najbardziej podobnych klastrów
    for (int i = 0; i < clusters.size(); i++) {
        for (int j = i + 1; j < clusters.size(); j++) {
            double similarity =
                this.similarityMeasure.computeSimilarity(clusters.get(i),
                    clusters.get(j));
            if (similarity > clustersToJoinSimilarity) {
                clustersToJoinSimilarity = similarity;
                firstClusterToJoinNumber = i;
                secondClusterToJoinNumber = j;
            }
        }
    }
    // połącz najbardziej podobne klastry
    clusters.get(firstClusterToJoinNumber).getDocuments()
        .addAll(clusters.get(secondClusterToJoinNumber).getDocuments());
    clusters.remove(secondClusterToJoinNumber);
}
return clusters;
}}

```

Listing 4.2: Algorytm grupowania algomerycyjnego — wariant HAC-Threshold

```

public class HACAlgorithmWithThreshold implements
    ClusteringAlgorithm {
    // (Tu występuje deklaracja pól i konstruktorów klasy, usnięta dla
    // przejrzystości)
    @Override
    public Collection<Cluster> cluster(Map<String, Object> params)
        throws WrongParamException {
        // Pobranie parametrów grupowania – progę, przy którym klastry

```

```

    przestają być łączone
    if (!params.containsKey(JOIN_THRESHOLD_PARAM)) {
        throw new WrongParamException("Join threshold must be provided
            as parameter!");
    }
    double joinThreshold = (Double) params.get(JOIN_THRESHOLD_PARAM);

    // Utwórz początkowe jednodokumentowe klastry
    List<Cluster> clusters = new
        ArrayList<Cluster>(corpus.getDocuments().size());
    for (Document d : corpus.getDocuments()) {
        clusters.add(new Cluster(new
            ArrayList<Document>(Arrays.asList(d))));
    }

    // Łącz klastry póki to możliwe
    while (clusters.size() > 1) {
        double clustersToJoinSimilarity = -1d;
        int firstClusterToJoinNumber = 0, secondClusterToJoinNumber = 0;
        // Znajdź parę najbardziej podobnych klastrów
        for (int i = 0; i < clusters.size(); i++) {
            for (int j = i + 1; j < clusters.size(); j++) {
                double similarity =
                    this.similarityMeasure.computeSimilarity(clusters.get(i),
                        clusters.get(j));
                if (similarity > clustersToJoinSimilarity) {
                    clustersToJoinSimilarity = similarity;
                    firstClusterToJoinNumber = i;
                    secondClusterToJoinNumber = j;
                }
            }
        }
        // Jeżeli podobieństwo najbliższych klastrów jest mniejsze od
        // progu łączenia – zakończ scalanie
        if (clustersToJoinSimilarity < joinThreshold) {
            break;
        }
        // Połącz najbardziej podobne klastry
        clusters.get(firstClusterToJoinNumber).getDocuments()
            .addAll(clusters.get(secondClusterToJoinNumber).getDocuments());
        clusters.remove(secondClusterToJoinNumber);
    }

    // Scal wszystkie jednodokumentowe klastry w jeden i oznacz jako
    // niepowiązane z żadnym innym dokumentem
    List<Document> lonelyDocuments = new ArrayList<Document>();
    Iterator<Cluster> clusterIterator = clusters.iterator();
    while (clusterIterator.hasNext()) {
        Cluster currentCluster = clusterIterator.next();
    }

```

```

    if (currentCluster.getDocuments().size() == 1) {
        lonelyDocuments.addAll(currentCluster.getDocuments());
        clusterIterator.remove();
    }
}
if (lonelyDocuments.size() > 0) {
    Cluster otherDocsCluster = new Cluster(lonelyDocuments);
    otherDocsCluster.setLabel(ClusterLabeller.OTHERS_LABEL);
    clusters.add(otherDocsCluster);
}
return clusters;
}
}

```

### 4.3 Prezentacja wyników

Ponieważ celem grupowania dokumentów jest ułatwienie człowiekowi zapoznania się z kolekcją tekstów, istotną kwestią jest właściwa prezentacja wyników. Ekran wyników Ugrupatora składa się z trzech części (rysunek 5): nazw zidentyfikowanych klastrów (w postaci płaskiej listy lub drzewa binarnego), spisu dokumentów znajdujących się w wybranym klastrze i treści aktualnie przeglądano dokumentu.

Rysunek 5 Ekran wyników grupowania



Zastosowane zostały dwa podejścia do nadawania nazw grupom. W pierwszym z nich etykietą klastra zostaje tytuł jego medoidu (założeniem jest, że nazwa pliku z tekstem jest jednocześnie jego tytułem). Dzięki temu nazwa klastra jest zwykle deskryptywna i zrozumiała dla użytkownika. Opisuje ona dobrze zawartość

Tabela 4.1: Porównanie różnych sposobów nadawania etykiet klastrom: słowa kluczowe i tytuł medoidu. Konfiguracja ważenia terminów: nazwy własne: 4, frazy: 3, rzeczowniki: 2, przymiotniki: 1, waga  $w_{tf}$ , bez użycia WordNetu

Nr	Słowa kluczowe	Tytuł medoidu
1	[wyciek ropy, bp, ropa, plama ropy, dno oceanu]	Pięć sposobów bp na plamę ropy
2	[PZU, skarb państwa, akcja, prezes, spółka]	Kurs PZU lekko w dół. Bank UBS wycenia je na 395 zł
3	[samolot, prezydent, lotnisko, Smoleńsk, katastrofa]	Lech Kaczyński nie żyje. Katastrofa samolotu. 96 ofiar
4	[Libia, członek załogi, katastrofa, samolot, Trypolis]	Libia. W katastrofie samolotu zginęły 103 osoby.
5	[radio, Rydzyk, Radio Maryja, ojciec, strona]	Ojciec Rydzyk uruchomił nowe radio

Tabela 4.2: Porównanie różnych sposobów nadawania etykiet klastrom, od wyników pokazanych w tabeli 4.1 różni się tym, że tutaj grupy o katastrofach lotniczych w Smoleńsku i Libii zostały scalone.

Nr	Słowa kluczowe	Tytuł medoidu
3+4	[samolot, katastrofa, lotnisko, Libia, prezydent]	Lech Kaczyński nie żyje. Katastrofa samolotu. 96 ofiar

grupy pod warunkiem, że znajdujące się w niej dokumenty są bardzo silnie powiązane — na przykład są to artykuły prasowe różnych wydawców opisujące to samo wydarzenie. Nie zawsze się ona jednak dobrze sprawdza przy grupach nieco słabiej powiązanych. Wówczas lepszą etykietą może być zbiór słów kluczowych, wybranych spośród terminów tworzących reprezentację dokumentów (tabela 4.2 pokazuje etykiety dla niejednorodnej grupy — tytuł medoidu pasuje tylko do niektórych artykułów, zaś słowa kluczowe opisują zawartość całości)

Wybór odpowiednich słów kluczowych może opierać się na porównywaniu częstości występowania konkretnych terminów w klastrze i poza nim, testy jednak wykazały, że bardzo dobrze sprawdza się trywialna metoda polegająca na wyborze kilku (ok. 5) terminów o największej wadze we wszystkich dokumentach z grupy (wynika to z zaawansowanych metod ważenia terminów, dzięki którym najistotniejsze i najbardziej treściwe leksemy i frazy mają zawsze największe wagi i one mają największy wpływ na grupowanie).

## Rozdział 5

---

# Badanie efektywności grupowania dokumentów

---

### 5.1 Ocena reprezentacji dokumentów

#### 5.1.1 Cel i metoda ewaluacji eksperymentu

Efekt działania algorytmów grupowania w bardzo dużej mierze zależy od tego, na ile skutecznie będzie obliczane podobieństwo pomiędzy dokumentami. Z tego powodu bardzo istotne jest stworzenie jak najlepszych reprezentacji dokumentów i to one będą tematem niniejszego eksperymentu.

Eksperyment ma na celu pokazanie, jaki wpływ na reprezentację dokumentów i wartości podobieństwa między nimi mają następujące metody:

- lepsze wstępne przetworzenie tekstu (tokenizacja i lematyzacja)
- selekcja poszczególnych części mowy
- rozpoznawanie fraz i nazw własnych
- wykorzystanie WordNetu
- różne sposoby ważenia terminów ( $w_{tf}$  i  $w_{tf-idf}$ )

Ponieważ nie znalazłem w literaturze poświęconej grupowaniu dokumentów metody oceny reprezentacji dokumentów (klasyczne metody oceniają cały proces grupowania), opracowałem własny sposób. Opiera się on na założeniu, że dokumenty mają dobrą reprezentację, jeżeli wartość podobieństwa  $sim_{cosine}$  pomiędzy dwoma dokumentami na ten sam temat (z tego samego klastra) jest duża, zaś dla dokumentów na różne tematy (z różnych klastrów) - mała. Do badania jakości

reprezentacji wprowadzę pojęcie **spójności** podziału kolekcji na grupy. Spójność będzie obliczana przy użyciu następujących wartości:

- minimalne podobieństwo między parą *przyległych* dokumentów, czyli takich, że nie istnieje trzeci dokument w tej samej grupie, który jest *pomiędzy* tymi dokumentami, tzn. jest bardziej podobny do nich obu niż one do siebie (większa wartość oznacza lepszą spójność):

$$A(d_1, d_2) \Leftrightarrow \exists_{g \in G} (d_1 \in g \wedge d_2 \in g \wedge \neg \exists_{d_3 \in g} (sim(d_1, d_3) < sim(d_1, d_2) \wedge sim(d_2, d_3) < sim(d_1, d_2)))$$

$$sim_{min-same} = \min_{d_1 \in C, d_2 \in C} \{sim(d_1, d_2) : A(d_1, d_2)\}$$

- średnie podobieństwo między dwoma dokumentami należącymi do tej samej grupy (większa wartość oznacza lepszą spójność):

$$sim_{avg-same} = \frac{1}{\sum_{g \in G} (|g| \cdot (|g| - 1))} \sum_{g \in G} \sum_{d_1 \in g} \sum_{d_2 \in g - \{d_1\}} sim(d_1, d_2)$$

- maksymalne podobieństwo między dwoma dokumentami z różnych grup (mniejsza wartość oznacza lepszą spójność):

$$sim_{max-distinct} = \max\{sim(d_1, d_2) : d_1 \in g_1 \wedge d_2 \in g_2 \wedge g_1 \in G \wedge g_2 \in G \wedge g_1 \neq g_2\}$$

- średnie podobieństwo między dwoma dokumentami z różnych grup (mniejsza wartość oznacza lepszą spójność):

$$sim_{avg-distinct} = \frac{1}{\sum_{g_1 \in G} \sum_{g_2 \in G - \{g_1\}} |g_1| \cdot |g_2|} \sum_{g_1 \in G} \sum_{g_2 \in G - \{g_1\}} sim(d_1, d_2)$$

Spójność grupowania może być badana pod kątem wartości średnich lub skrajnych, można zatem zdefiniować dwie wartości oceniające podział:

$$coh_1 = sim_{avg-same} - sim_{avg-distinct}$$

$$coh_2 = sim_{min-same} - sim_{max-distinct}$$

Obie zdefiniowane powyżej miary przyjmują wartości z zakresu  $\langle -1, 1 \rangle$ . Pierwsza z nich opiera się na średnich podobieństwach, jej większa wartość poprawi wyniki grupowania np. algorytmem aglomeracyjnym z funkcją podobieństwa  $SIM_{group-average}$ . Z kolei miara  $coh_2$  zależy od wartości skrajnych (minimalnej i maksymalnej) i wskazywać będzie warunki dla algorytmu DBSCAN lub aglomeracyjnego z funkcją  $SIM_{single-linkage}$ . Zauważmy, że jeżeli  $coh_2 > 0$ , to dla

danej kolekcji dokumentów istnieje pewna wartość podobieństwa  $sim_0$  taka, że poprawne klastry są klasami abstrakcji relacji

$$R_{sim}(d_1, d_2) \Leftrightarrow sim(d_1, d_2) > sim_0$$

Aby połączyć właściwości współczynników  $coh_1$  i  $coh_2$ , wprowadzę miarę  $coh_3$ , uwzględniającą w trywialny sposób ocenę podobieństw średnich i skrajnych:

$$coh_3 = coh_1 + coh_2$$

Wyznacznikiem jakości reprezentacji dokumentów będzie wartość spójności *prawidłowego* podziału kolekcji na grupy. Prawidłowy przydział dokumentów do poszczególnych grup jest tworzony ręcznie. Wyższa spójność takiego grupowania będzie interpretowana jako lepsza reprezentacja dokumentów, gdyż oznacza większe wartości podobieństwa między dokumentami podobnymi i mniejsze między dokumentami niepodobnymi.

### 5.1.2 Korpus testowy

Do testów jakości reprezentacji dokumentów posłużył ręcznie skonstruowany korpus **K1**, zawierający 33 artykuły o bieżących wydarzeniach, pochodzące z ogólnodostępnych informacyjnych portali internetowych takich jak *onet.pl*, *gazeta.pl* czy *rp.pl*. Tytuły i źródła poszczególnych tekstów opisane są w dodatku **B**. Zostały one manualnie pogrupowane na następujące klastry tematyczne:

1. wyciek ropy na platformie BP w Zatoce Meksykańskiej (8 dokumentów)
2. debiut akcji PZU na giełdzie (3 dokumenty)
3. katastrofa samolotu z Lechem Kaczyńskim w Smoleńsku (3 dokumenty)
4. katastrofa samolotu w Libii (2 dokumenty)
5. uruchomienie radia internetowego SIM przez Tadeusza Rydzyska (3 dokumenty)
6. powódź na wrocławskim osiedlu Kozanów (3 dokumenty)
7. skazanie Anny Jaruckiej za składanie fałszywych zeznań (5 dokumentów)
8. przyznanie Złotych Lwów na festiwalu w Gdyni (6 dokumentów)

W powyżej opisanym korpusie interpretacja wartości podobieństwa pomiędzy dokumentami mówiącymi o katastrofie samolotu w Libii a tymi o tragedii w Smoleńsku pozostaje niejednoznaczna: przedstawiają one różne wydarzenia, więc w naszym podziale stanowią różne grupy i pożądane byłyby niskie wartości podobieństwa między nimi. Z drugiej jednak strony, są one bardzo zbliżone

tematyczne, więc funkcja podobieństwa powinna to uwidoczniać. Niestety, grupowanie dokumentów jest pełne takich wieloznaczności, a to, czy dane dokumenty powinny być w jednym klastrze, zależy często od kontekstu. Aby jednak uzyskane wyniki były bardziej miarodajne, badania spójności podziału zostały powtórzone również dla zmodyfikowanego korpusu **K2**, w którym usunięte zostały dwa artykuły o wypadku libijskim, co pozwoliło uzyskać bardziej czytelny i jednoznaczny podział.

### 5.1.3 Wyniki eksperymentu i wnioski

#### Ocena wpływu różnych części mowy i schematu ważenia na reprezentację dokumentów

Pierwsza seria testów jakości reprezentacji dokumentów została przeprowadzona w celu zbadania znaczenia różnych części mowy oraz wpływu sposobu ważenia terminów na spójność grupowania. W badaniach porównano reprezentację standardową oraz rozszerzoną przy uwzględnieniu rzeczowników, przymiotników, czasowników i przysłówków, dla których wagi obliczane są za pomocą funkcji  $w_{tf}$  lub  $w_{tf-idf}$ . WordNet na tym etapie nie jest jeszcze wykorzystany. Wyniki testów dla poszczególnych korpusów prezentują tabele 5.1 i 5.2.

Pierwszym wnioskiem, jaki wynika z analizy wyników badań, jest to, iż spójność reprezentacji standardowej jest zauważalnie gorsza od reprezentacji opartej na wynikach działania parsera Translatiki. Różnica ta pokazuje, w jakim stopniu można polepszyć reprezentację poprzez bardziej zaawansowaną tokenizację, rozpoznawanie fraz oraz odrzucanie wszystkich spójników, zaimków oraz przymioków (nie tylko tych zawartych na liście słów funkcyjnych).

Zauważyć również można, iż prosta waga  $w_{tf}$ , zliczająca jedynie wystąpienia terminów w dokumencie, sprawdziła się lepiej (w sensie współczynników  $coh_1$  i  $coh_3$ ) od wagi  $w_{tf-idf}$ , korygującej wagę terminu o jego frekwencję w całym korpusie. Uzasadnić można to faktem, iż w rozpatrywanej kolekcji istotne wyrazy powtarzały się głównie w dokumentach mówiących o tym samym wydarzeniu. Ze względu na małą liczbę analizowanych tekstów, waga  $w_{tf-idf}$  zaniżała wartość słów kluczowych, które współwystępowały w artykułach na ten sam temat, przez co reprezentacja ją wykorzystująca była zwykle nieco gorsza (efekty tego zjawiska można zobaczyć w tabeli 5.3). Wykorzystanie wagi  $w_{tf-idf}$  polepsza jednak współczynniki  $coh_2$ , zwłaszcza w przypadku korpusu K1. Należy się spodziewać, że w dużych kolekcjach dokumentów z tej samej dziedziny  $w_{tf-idf}$  umożliwi lepsze wydatnienie terminów kluczowych.

Trzecim wnioskiem jest to, że pozytywny wpływ na reprezentację dokumentu mają jedynie rzeczowniki - dodanie przymiotników, czasowników czy przysłówków niemal zawsze pogarsza spójność grupowania (pomijalnie mały zysk na współczynniku  $coh_2$  przyniosło jedynie dodanie przymiotników w kolekcji K1).



Tabela 5.1: Wyniki badania spójności grupowania - korpus K1

Oznaczenia: w - sposób ważenia; N - użycie rzeczownika; A - użycie przymiotnika;  
V - użycie czasownika; Aj - użycie przysłówka

konfiguracja					sim				spójność		
w	N	A	V	Aj	avg-dis	avg-same	max-dis	min-same	$coh_1$	$coh_2$	$coh_3$
tf-idf	rep. standard				0,055	0,525	0,525	0,288	0,470	-0,237	0,233
tf-idf	1	1	1	1	0,045	0,522	0,462	0,209	0,477	-0,253	0,224
tf-idf	1	0	0	0	0,041	0,600	0,485	0,273	0,559	-0,212	0,348
tf-idf	1	1	0	0	0,043	0,575	0,468	0,249	0,532	-0,219	0,313
tf-idf	1	0	1	0	0,042	0,544	0,495	0,236	0,502	-0,258	0,243
tf-idf	1	0	0	1	0,042	0,586	0,469	0,245	0,544	-0,224	0,319
tf	rep. standard				0,264	0,694	0,701	0,485	0,430	-0,216	0,214
tf	1	1	1	1	0,080	0,667	0,632	0,322	0,587	-0,309	0,278
tf	1	0	0	0	0,068	0,721	0,705	0,400	0,653	-0,306	0,348
tf	1	1	0	0	0,074	0,704	0,665	0,372	0,630	-0,293	0,337
tf	1	0	1	0	0,073	0,686	0,658	0,351	0,612	-0,307	0,306
tf	1	0	0	1	0,070	0,711	0,689	0,374	0,641	-0,316	0,325

Tabela 5.2: Wyniki badania spójności grupowania - korpus K2

Oznaczenia: w - sposób ważenia; N - użycie rzeczownika; A - użycie przymiotnika;  
V - użycie czasownika; Aj - użycie przysłówka

konfiguracja					sim				spójność		
w	N	A	V	Aj	avg-dis	avg-same	max-dis	min-same	$coh_1$	$coh_2$	$coh_3$
tf-idf	rep. standard				0,050	0,520	0,261	0,282	0,471	0,021	0,492
tf-idf	1	1	1	1	0,041	0,518	0,198	0,205	0,477	0,008	0,484
tf-idf	1	0	0	0	0,035	0,596	0,229	0,269	0,562	0,040	0,602
tf-idf	1	1	0	0	0,038	0,571	0,235	0,245	0,533	0,010	0,543
tf-idf	1	0	1	0	0,037	0,540	0,210	0,232	0,503	0,021	0,525
tf-idf	1	0	0	1	0,037	0,582	0,226	0,241	0,545	0,015	0,560
tf	rep. standard				0,263	0,694	0,598	0,485	0,431	-0,113	0,318
tf	1	1	1	1	0,075	0,667	0,357	0,322	0,592	-0,035	0,558
tf	1	0	0	0	0,060	0,721	0,364	0,400	0,661	0,035	0,696
tf	1	1	0	0	0,068	0,704	0,400	0,372	0,636	-0,028	0,608
tf	1	0	1	0	0,067	0,686	0,339	0,351	0,619	0,012	0,631
tf	1	0	0	1	0,063	0,711	0,360	0,374	0,648	0,014	0,662

Tabela 5.3: 5 najwyżej ważonych terminów w dokumencie *Firma BP: Do wtorku chcemy zatamować wyciek ropy* przy metodzie ważenia  $w_{tf}$  i  $w_{tf-idf}$ 

Termin	$w_{tf}$	Termin	$w_{tf-idf}$
ropa	0,105	ropa	0,069
wyciek	0,053	wtorek	0,039
plama ropy	0,035	wyciek	0,035
bp	0,035	plama ropy	0,031
wyciek ropy	0,035	plama	0,031

### Ocena wpływu WordNetu, fraz i nazw własnych na reprezentację dokumentów

Druga seria testów miała na celu pokazanie wpływu wykorzystania WordNetu w obliczaniu podobieństwa między dokumentami oraz znalezienie odpowiednich współczynników wag dla fraz, nazw własnych i zwykłych jednowyrazowych leksemów. Zgodnie z wnioskami uzyskanymi w poprzednim eksperymencie, pod uwagę brane były jedynie rzeczowniki, a terminy ważne funkcją  $w_{tf}$ . Wyniki zaprezentowane są w tabelach 5.4 i 5.5.

Wyniki pokazują, że użycie WordNetu ma niewielki wpływ na spójność grupowania. W dużej mierze jest to skutkiem sposobu rozpoznawania synsetów, który umożliwia bardzo wysoką skuteczność poprawnego semantycznie przypisania wyrazu do synsetu, lecz ignoruje wystąpienia synonimów, które są niepewne. Takie rygorystyczne podejście było jednak jedynym, które przyniosło choć w części pozytywne rezultaty, rozpoznając np. jednakowe znaczenie słów  $\{film, produkcja, obraz\}$  w jednym dokumencie oraz  $\{film, obraz\}$  w drugim. W korpusie K2 użycie WordNetu poprawiło współczynnik  $coh_3$  średnio o ok. 0,01, co jest raczej nieznaczącą zmianą. W przypadku korpusu K1 wykorzystanie WordNetu pogorszyło nieco spójność grupowania. Wynika to z zauważalnie większego wzrostu podobieństwa  $sim_{max-dis}$  (średnio o 0,05) w stosunku do wzrostu innych podobieństw (średnio o 0,01), czego efektem było pogorszenie się spójności  $coh_2$  i  $coh_3$ . Łatwo znaleźć wytłumaczenie tego faktu - zwiększyła się wartość podobieństwa pomiędzy dokumentami o katastrofie lotniczej w Smoleńsku i w Libii. To właśnie w przypadku takich zbliżonych (ale nie identycznych) tematycznie dokumentów WordNet może wyszukać dużo podobieństw, które nie zostałyby znalezione inną metodą.

Innym wnioskiem wynikającym z tego eksperymentu jest fakt, iż przypisanie większych wag nazwom własnym ma pozytywny wpływ na jakość reprezentacji. Zależność ta była możliwa do przewidzenia, nazwy silnie wpływają na tematykę dokumentów i obliczanie podobieństwa w oparciu głównie o ich wystąpienia ma zwykle pozytywny skutek. W korpusie K1 najlepsze wyniki można było uzyskać

Tabela 5.4: Wyniki badania spójności grupowania - korpus K1, posortowane malejąco względem współczynnika  $coh_3$ 

Oznaczenia: WN - użycie WordNetu; PN - waga dla nazw własnych; PH - waga dla fraz

konfiguracja			sim				spójność		
WN	PN	PH	avg-dis	avg-same	max-dis	min-same	$coh_1$	$coh_2$	$coh_3$
-	3	0	0,049	0,695	0,486	0,455	0,646	-0,031	0,615
-	3	1	0,048	0,692	0,480	0,449	0,644	-0,030	0,613
-	3	2	0,045	0,684	0,463	0,435	0,639	-0,028	0,611
-	3	3	0,042	0,676	0,437	0,412	0,634	-0,024	0,610
+	3	0	0,058	0,701	0,523	0,464	0,643	-0,059	0,584
+	3	1	0,057	0,700	0,519	0,459	0,643	-0,060	0,583
+	3	2	0,053	0,694	0,503	0,444	0,641	-0,059	0,582
+	3	3	0,049	0,687	0,479	0,422	0,638	-0,058	0,581
-	2	3	0,047	0,685	0,484	0,375	0,638	-0,109	0,528
-	2	2	0,053	0,696	0,533	0,406	0,643	-0,127	0,516
-	2	1	0,057	0,706	0,572	0,427	0,649	-0,145	0,504
-	2	0	0,059	0,712	0,586	0,435	0,653	-0,152	0,501
+	2	3	0,057	0,698	0,569	0,390	0,641	-0,178	0,463
+	2	1	0,069	0,717	0,631	0,443	0,649	-0,188	0,461
+	2	2	0,063	0,709	0,606	0,422	0,645	-0,184	0,461
+	2	0	0,071	0,720	0,639	0,450	0,649	-0,189	0,460
-	1	3	0,052	0,691	0,567	0,329	0,639	-0,238	0,401
-	1	2	0,060	0,706	0,646	0,370	0,646	-0,276	0,369
+	1	3	0,064	0,706	0,640	0,354	0,642	-0,286	0,356
+	1	1	0,082	0,734	0,725	0,425	0,652	-0,299	0,353
+	1	2	0,073	0,721	0,690	0,395	0,648	-0,295	0,353
-	1	1	0,068	0,721	0,705	0,400	0,653	-0,306	0,348
+	1	0	0,085	0,739	0,744	0,437	0,654	-0,307	0,346
-	1	0	0,071	0,729	0,728	0,411	0,659	-0,317	0,342
-	0	3	0,054	0,694	0,605	0,305	0,639	-0,300	0,339
+	0	3	0,067	0,709	0,669	0,335	0,642	-0,333	0,309
+	0	2	0,078	0,727	0,730	0,382	0,649	-0,347	0,302
-	0	2	0,064	0,711	0,696	0,350	0,647	-0,346	0,301
+	0	1	0,088	0,743	0,789	0,417	0,655	-0,372	0,283
+	0	0	0,092	0,749	0,811	0,431	0,657	-0,380	0,276
-	0	1	0,072	0,728	0,765	0,385	0,656	-0,380	0,276
-	0	0	0,076	0,738	0,791	0,398	0,662	-0,393	0,269

Tabela 5.5: Wyniki badania spójności grupowania - korpus K2, posortowane malejąco względem współczynnika  $coh_3$ 

Oznaczenia: WN - użycie WordNetu; PN - waga dla nazw własnych; PH - waga dla fraz

konfiguracja			sim				spójność		
WN	PN	PH	avg-dis	avg-same	max-dis	min-same	$coh_1$	$coh_2$	$coh_3$
+	2	1	0,063	0,719	0,393	0,443	0,656	0,050	0,706
+	2	2	0,058	0,710	0,368	0,422	0,653	0,053	0,706
+	2	0	0,065	0,721	0,402	0,450	0,656	0,049	0,705
+	2	3	0,051	0,699	0,334	0,390	0,648	0,056	0,705
-	1	0	0,063	0,729	0,380	0,411	0,666	0,031	0,698
-	1	1	0,060	0,721	0,364	0,400	0,661	0,035	0,696
-	2	0	0,054	0,714	0,402	0,435	0,660	0,033	0,693
-	1	2	0,053	0,706	0,332	0,370	0,653	0,038	0,691
-	2	1	0,052	0,708	0,393	0,427	0,656	0,035	0,690
-	2	2	0,048	0,697	0,368	0,406	0,649	0,038	0,687
-	1	3	0,046	0,691	0,291	0,329	0,646	0,038	0,684
-	2	3	0,042	0,686	0,334	0,375	0,644	0,041	0,684
+	1	2	0,066	0,721	0,377	0,395	0,656	0,018	0,674
+	1	3	0,056	0,706	0,330	0,354	0,650	0,024	0,674
+	1	1	0,074	0,734	0,417	0,425	0,660	0,008	0,669
+	1	0	0,077	0,739	0,432	0,437	0,662	0,004	0,666
-	0	2	0,056	0,710	0,371	0,350	0,654	-0,021	0,634
-	0	3	0,047	0,693	0,321	0,305	0,646	-0,016	0,630
+	3	3	0,045	0,690	0,437	0,422	0,644	-0,015	0,629
-	0	1	0,064	0,727	0,421	0,385	0,663	-0,035	0,628
+	3	1	0,053	0,702	0,480	0,459	0,649	-0,021	0,628
+	3	2	0,050	0,697	0,463	0,444	0,647	-0,019	0,628
-	0	0	0,067	0,737	0,441	0,398	0,670	-0,043	0,627
+	3	0	0,054	0,703	0,486	0,464	0,649	-0,022	0,627
+	0	3	0,058	0,708	0,362	0,335	0,650	-0,026	0,624
-	3	0	0,046	0,698	0,486	0,455	0,651	-0,031	0,620
-	3	1	0,045	0,694	0,480	0,449	0,649	-0,030	0,618
+	0	2	0,069	0,726	0,423	0,382	0,657	-0,041	0,616
-	3	2	0,042	0,686	0,463	0,435	0,644	-0,028	0,616
-	3	3	0,039	0,678	0,437	0,412	0,640	-0,024	0,615
+	0	1	0,079	0,741	0,475	0,417	0,662	-0,057	0,605
+	0	0	0,083	0,748	0,494	0,431	0,665	-0,064	0,601

zwiększając trzykrotnie wagę każdej nazwy własnej. Co ciekawe, wartości  $coh_3$  w obu korpusach przy tak ustawionych wagach są bardzo podobne - różnią się o około 0,01. Zmniejszanie tej wagi powoduje istotne zwiększenie podobieństwa  $sim_{max-dis}$  w K1 - jak wiemy, podobieństwo to dotyczy dokumentów o katastrofie lotniczej, słownictwo je opisujące jest zbliżone, lecz nazwy własne się tam nie powtarzają, zatem zwiększenie wagi dla tych ostatnich podkreśla różnice między tymi dokumentami, zmniejszając wartość podobieństwa między nimi. Analiza wyników w K2 sugeruje nieco niższą wartość wagi dla nazw własnych — około 1-2, lecz w tym przypadku nie ma to tak silnego przełożenia na spójność.

Dość zaskakującą właściwością jest fakt, że zwiększenie wagi dla fraz nie ma zbyt dużego znaczenia dla wartości spójności grupowania, a w niektórych przypadkach nawet pomijanie fraz pomaga polepszyć reprezentację (należy tu przypomnieć, że dotyczy to jedynie fraz nie będących nazwami własnymi, gdyż te są traktowane zawsze osobno). Wyjaśnieniem dla tej sytuacji jest prawdopodobnie to, że leksemy składowe fraz są niezależnie dodawane do reprezentacji.

## 5.2 Ocena algorytmów grupowania

### 5.2.1 Metoda ewaluacji eksperymentu

Do ewaluacji grupowania dokumentów powszechnie używanych jest kilka miar, które zostaną przytoczone za [MRS08]. Jedną z nich jest **czystość** (ang. *purity*), która oznacza odsetek dokumentów, które zostały prawidłowo zaklasyfikowane. Dla grupowania  $G = \{g_1, g_2, \dots, g_k\}$  kolekcji dokumentów  $C = \{d_1, d_2, \dots, d_m\}$ , dla której prawidłowym podziałem jest  $G^* = \{g_1^*, g_2^*, \dots, g_j^*\}$ , czystość zdefiniowana jest następująco:

$$purity(G) = \frac{1}{m} \sum_k \max_j |g_k \cap g_j^*|$$

Miara ta może być jednak wykorzystana jedynie wówczas, gdy znane jest prawidłowe grupowanie kolekcji dokumentów, a to musi zostać wykonane manualnie, co jest niezwykle pracochłonne dla dużych nieuporządkowanych kolekcji.

Inną miarą oceny grupowania dokumentów jest **indeks Randa** (RI). Ocenia on, jaka część decyzji o przypisaniu dokumentów do poszczególnych grup jest prawidłowa. Dla potrzeb obliczenia tego wskaźnika przyjmuje się, że grupowanie polega na podjęciu jednej decyzji w stosunku do każdej pary dokumentów, które mogą być:

**prawdziwie pozytywne** (*TP, true positive*) - dwa podobne dokumenty zostały przypisane do tej samej grupy

**prawdziwie negatywne** (*TN, true negative*) - dwa różne dokumenty zostały przypisane do różnych grup

**falszywie pozytywne** (*FP, false positive*) - dwa różne dokumenty zostały przypisane do tej samej grupy

**falszywie negatywne** (*FN, false negative*) - dwa podobne dokumenty zostały przypisane do różnych grup

Indeks Randa jest stosunkiem liczby poprawnych decyzji do liczby wszystkich decyzji, których jest  $\frac{m(m-1)}{2}$  (dla  $m$  dokumentów)”

$$RI(G) = \frac{\#TP + \#TN}{\#TP + \#FP + \#FN + \#TN} = \frac{\#TP + \#TN}{\frac{m(m-1)}{2}}$$

Wartością łatwiejszą do obliczenia, gdyż wymagającą jedynie analizowania podobieństwa dokumentów wewnątrz poszczególnych klastrów (bez porównań między klastrami), jest **precyzja**, czyli stosunek liczby par podobnych dokumentów do liczby wszystkich par w grupach:

$$precision(G) = \frac{\#TP}{\#TP + \#FP} = \frac{\#TP}{\sum_{g \in G} \binom{|g|}{2}}$$

Ponieważ nie wszystkie oceniane algorytmy grupują całą kolekcję (DBSCAN i HAC-Threshold mogą pozostawiać dokumenty niepowiązane z innymi), wyniki grupowania będą dodatkowo zawierały informację, jaka część całego zbioru dokumentów została przydzielona do grup — wartość tę będę nazywał **pokryciem**. Pokrycie nie stanowi niezależnej oceny wyniku algorytmu, gdyż nie jest zależne od prawdziwych powiązań między dokumentami, umożliwia ono jednak porównanie działania różnych algorytmów dla tej samej kolekcji dokumentów.

### 5.2.2 Opis eksperymentu

Eksperyment składa się z dwóch części. Pierwsza z nich polega na zastosowaniu algorytmów  $k$ -średnich, DBSCAN oraz hierarchicznych (za wyjątkiem HAC zwracającego wynik w postaci drzewa, gdyż ten jest trudno ocenić) do podziału korpusu K1. Jego wyniki będą oceniane wszystkimi miarami opisanymi w 5.2.1.

W drugiej części eksperymentu użyte zostały artykuły z Korpusu Rzeczpospolitej ([rze]). Ponieważ są one przechowywane w formie plików html, konieczne było napisanie pomocniczego programu ekstrahującego z nich treść tekstową. Został on zaimplementowany w oparciu o bibliotekę Jericho HTML Parser<sup>1</sup>. Napisany program, oprócz ekstrakcji tekstu, pobiera informację o dziale, tytule i podtytule artykułu z meta tagów pliku i tworzy z nich nową opisową nazwę dla pliku.

Z tego korpusu wybranych zostało 205 artykułów, które zostały opublikowane w marcu 2002 roku w działach Kraj oraz Świat. Razem z 33 tekstami

<sup>1</sup>Jest to biblioteka z otwartym źródłem w Javie, strona projektu: <http://jericho.htmlparser.net/docs/index.html> (dostęp: czerwiec 2010)

z korpusu K1 (zostały one dołączone, by można było sprawdzić, czy predefiniowane grupy zostaną również wykryte wśród innych dokumentów) stanowią one kolekcję testową, która w dalszej części pracy będzie oznaczana jako **K3**. Ponieważ w tym zbiorze dużo dokumentów nie jest powiązanych z żadnym innym, tworzenie podziału obejmującego wszystkie artykuły byłoby sztuczne - stworzone powinny zostać tylko grupy łączące faktycznie podobne tematycznie artykuły. Dlatego testy na tym korpusie wykonane zostaną jedynie dla algorytmów grupowania częściowego, czyli DBSCAN oraz HAC-Threshold. Parametry wejściowe dla tych algorytmów zostaną dobrane eksperymentalnie. Ponieważ wzorcowe grupowanie dla tych artykułów nie istnieje, niemożliwe jest wykorzystanie do oceny wyników miary czystości oraz indeksu Randa. Obliczona będzie jedynie precyzja, opierająca się na ocenie zawartości każdej z uzyskanych grup.

### 5.2.3 Wyniki eksperymentu i wnioski

#### Część I — kolekcja K1

Wyniki pierwszej części eksperymentu zaprezentowane są w tabeli 5.6. Zgodnie z wcześniejszymi obawami, grupy uzyskane przez użycie algorytmu k-średnich nie są prawidłowe. Co więcej, początkowe losowanie sprawia, że kolejne wyniki uzyskiwane przez tę metodę nie są powtarzalne. Do oceny jakości grupowania został wybrany najlepszy wynik spośród otrzymanych w 5 wywołaniach algorytmu, nie jest on jednak całkowicie poprawny. Przykład 2 prezentuje błędnie rozpoznaną przez algorytm k-średnich grupę dokumentów, zawierającą dokumenty o dwóch katastrofach lotniczych oraz wycieku ropy w Zatoce Meksykańskiej (ten ostatni został uznany za podobny do poprzednich głównie z powodu powtarzającego się często słowa *katastrofa*).

*[katastrofa, samolot, Libia, lotnisko, Trypolis]*

1. Lech Kaczyński nie żyje. Katastrofa samolotu. 96 ofiar
2. Libia W katastrofie samolotu zginęły 103 osoby
3. Katastrofa platformy: plama ropy zagraża Ameryce
4. Nie żyje prezydencka para, politycy, żołnierze
5. Katastrofa samolotu w Libii
6. Głowa państwa polskiego nie żyje

**Przykład 2:** Błędna grupa rozpoznana przez algorytm k-średnich

Grupowania uzyskane w wyniku działania pozostałych algorytmów są lepsze. Algorytm aglomeracyjny HAC-Flat poprawnie przydzielił dokumenty do ośmiu

Tabela 5.6: Ocena wyników grupowania — korpus K1  
 konfiguracja: sposób ważenia  $w_{tf}$ ; brak użycia WordNetu; waga dla nazw własnych: 3; waga dla fraz: 2; reprezentacja tylko rzeczowników

Algorytm	Konfiguracja	Czystość	Precyzja	Pokrycie
k-średnich	$k = 8$	88%	84%	100%
HAC-Flat	$k = 8$	100%	100%	100%
HAC-Threshold	próg łączenia: 0, 3	94%	92%	100%
HAC-Threshold	próg łączenia: 0, 39	100%	100%	100%
HAC-Threshold	próg łączenia: 0, 5	100%	100%	97%
DBSCAN	próg sąsiedztwa: 0, 5, MinPts=1	100%	100%	97%
DBSCAN	próg sąsiedztwa: 0, 4, MinPts=1	100%	68%	100%

klastrów. Ponadto, jeżeli wykonany zostanie on dla siedmiu klastrów, otrzymamy grupowanie ze scalonymi artykułami dotyczącymi katastrofy w Smoleńsku oraz Libii - dokładnie tak, jak tego oczekiwaliśmy. Oczywiście, takie same wyniki możemy otrzymać przy użyciu algorytmu HAC-Threshold z odpowiednio ustalonym progiem łączenia, uzyskując właściwe grupy dla progu równego 0,39. Dla zbyt niskiego progu połączone zostaną grupy o katastrofach lotniczych, dla zbyt wysokiego - jeden dokument nie zostanie przydzielony do żadnej grupy<sup>2</sup>. Analiza kolejnych wartości podobieństwa  $SIM_{group-average}$  (pokazanych w tabeli 5.7), przy których następuje scalanie grup, wskazuje, że to właśnie te ostatnie dwa przypadki są najbardziej prawdopodobnym wynikiem grupowania przez ten algorytm, kolejne łączenia grup występujące po nich powodują największe zmniejszenie wartości  $SIM_{group-average}$ .

Rezultaty możliwe do uzyskania przy użyciu algorytmu DBSCAN są nieco gorsze w stosunku do algorytmów hierarchicznych, co objawia się tym, że przy podanej konfiguracji tworzenia reprezentacji niemożliwe jest uzyskanie grupowania zgodnego z wzorcem. Dla progu sąsiedztwa ustalonego na wartość 0.5 klastry zawierają jednolite tematycznie dokumenty, lecz jeden artykuł nie został nigdzie przypisany (taka sama sytuacja, jak dla zbyt wysokiego progu łączenia w algorytmie HAC-Threshold). Okazuje się jednak, że dla progu sąsiedztwa równego 0.4 scalone w jedną zostały grupy o katastrofach lotniczych i skazaniu Anny Jarruckiej<sup>3</sup>. Zauważmy, że w tym przypadku czystość wynosi 100% — wynika to z faktu, że wszystkie tematycznie podobne dokumenty należą do tej samej grupy.

<sup>2</sup>Artykułem, który pozostaje bez przydziału, jest *Wyeliminowanie niewygodnego kandydata - bezkarne* z portalu *salon24.pl*. W przeciwieństwie do innych dokumentów z tej grupy, nie jest on opisem wydarzenia, lecz luźnym komentarzem do niego - stąd wynika niewielkie podobieństwo do innych dokumentów o skazaniu Anny Jarruckiej.

<sup>3</sup>Fakt ten, zaskakujący na pierwszy rzut oka, wyjaśnić można wysokim współczynnikiem ważenia dla nazw własnych oraz powtarzającym się w tych artykułach nazwiskiem *Kaczyński*



Tabela 5.7: Podobieństwa łączonych grup w algorytmach grupowania aglomeracyjnego użytego w kolekcji K1

Liczba grup	Grupy jednoelementowe	$SIM_{group-average}$	Różnica $SIM_{group-average}$
33	33	1	-
32	31	0,946	0,05
31	29	0,917	0,03
30	28	0,884	0,03
29	27	0,871	0,01
28	25	0,869	0
27	23	0,847	0,02
26	22	0,841	0,01
25	21	0,835	0,01
24	20	0,796	0,04
23	19	0,772	0,02
22	17	0,764	0,01
21	15	0,739	0,02
20	13	0,721	0,02
19	11	0,714	0,01
18	10	0,709	0
17	8	0,698	0,01
16	7	0,693	0,01
15	7	0,664	0,03
14	6	0,646	0,02
13	4	0,617	0,03
12	3	0,589	0,03
11	3	0,563	0,03
10	2	0,541	0,02
9	0	0,500	0,04
8	0	0,398	0,1
7	0	0,390	0,01
6	0	0,105	0,28
5	0	0,077	0,03
4	0	0,058	0,02
3	0	0,048	0,01
2	0	0,030	0,02
1	0	0,021	0,01

## Część II — kolekcja K3

W drugiej części eksperymentu przeprowadzono grupowanie na korpusie K3 algorytmem HAC-Threshold z progiem łączenia równym 0.5 oraz algorytmem DBSCAN z minimalną liczbą sąsiadów równą 2 i progiem podobieństwa równym 0.5. Wyniki są zaprezentowane w tabeli 5.8.

Tabela 5.8: Ocena wyników grupowania — korpus K3  
konfiguracja: sposób ważenia  $w_{tf}$ ; brak użycia WordNetu; waga dla nazw własnych: 3; waga dla fraz: 2; reprezentacja tylko rzeczowników

Algorytm	Konfiguracja	Precyzja	Pokrycie	Grupy <sup>a</sup>
HAC-Threshold	próg łączenia: 0, 5	94%	67%	35/43
DBSCAN	próg sąsiedztwa: 0, 5, MinPts=1	88%	68%	32/41

<sup>a</sup>Format komórki: liczba w pełni poprawnie rozpoznanych grup / liczba wszystkich rozpoznanych grup

Każda uzyskana grupa została oceniona niezależnie przez 3 osoby pod względem podobieństwa znajdujących się w niej dokumentów. Jako "prawidłowa" ocena zawartości klastrów, która posłużyła do ewaluacji wyniku, została uznana taka, która się powtarzała w co najmniej dwóch ocenach testerów (w 60% wszystkie oceny były jednakowe, nie zdarzył się przypadek, w którym każda osoba wystawiła inną ocenę).

Wyniki działania algorytmu aglomeracyjnego są zadowalające. Większość (81%) klastrów została uznana za prawidłowe, pozostałe wprawdzie opisują różne wydarzenia, ale podobieństwo tematyczne między dokumentami w nich zawartymi jest łatwo dostrzegalne — przykład 3 prezentuje grupę, która zawiera trzy artykuły o finansowaniu szkół niepublicznych oraz jeden o likwidowaniu techników i przekształcaniu ich w licea ogólnokształcące. Największa spośród znalezionych grup zawierała 18 dokumentów omawiających konflikt na Bliskim Wschodzie (przykład 4).

*[szkoła, dotacja, NIK, uczeń, czesne]*

1. Dotacje zagrożone — Finansowanie szkół niepublicznych
2. Nie ma spisku — Niepaństwowe szkoły boją się zabrania dotacji
3. Wyższe dotacje, czesne bez zmian — NIK o szkołach niepublicznych
4. Powiaty zdecydowały, teraz pora na uczniów — W całym kraju zniknie niemal połowa techników, przybędzie 508 ogólniaków

**Przykład 3:** Błędna grupa rozpoznana przez algorytm hierarchiczny

Dodanie 205 artykułów z Rzeczpospolitej nie przeszkodziło we właściwym połączeniu dokumentów z klastra K1. Wśród 43 klastrów rozpoznanych w kolekcji

*[Arafat, Izrael, Palestyńczyk, Bliski Wschód, Szaron]*

1. Amerykański mediator rozpoczął misję — Wojska izraelskie wycofują się z Ramalli
2. Żniwo śmierci — Kolejne ofiary w Strefie Gazy i na Zachodnim Brzegu Jordanu — Amerykańskie mediacje
3. Szaron gotów do ustępstw — Amerykanie naciskają
4. Znowu fiasko — Kolejne spotkanie izraelsko-palestyńskiej komisji bezpieczeństwa
5. Wreszcie spokojniej — Armia izraelska opuszcza ziemie Autonomii Palestyńskiej — Arafat może wyjechać, ale nie wiadomo, czy będzie mógł wrócić
6. Znowu zamach w Izraelu — Prezydent USA rozczarowany sytuacją na Bliskim Wschodzie
7. Nie będzie łatwo — Misja Zinniego: Palestyńczycy i Izraelczycy wkrótce się spotkają
8. Nalot na siedzibę Arafata — Kolejny krwawy dzień w Izraelu i na terenach Autonomii Palestyńskiej
9. Niepewny los Arafata — Przywódca Palestyńczyków prawdopodobnie nie pojedzie na szczyt arabski do Bejrutu
10. Odliczanie przed Bejrutem — Kwestia wyjazdu Arafata na szczyt arabski nierozstrzygnięta
11. Zastrzelony z zimną krwią — Arafat może już opuszczać Ramallę
12. Atak za atakiem — Bez porozumienia w sprawie rozejmu
13. Krwawy odwet w Ziemi Świętej — W zamachach palestyńskich zginęło 21 osób — Izraelczycy użyli śmigłowców i czołgów
14. Powiedzieli Rzeczpospolitej — Yossi Melman, komentator izraelskiego dziennika "Haarec"
15. Krok w stronę rozejmu — Żołnierze wycofują się z Betlejem
16. Masakra pod Tel Awiwem — Saudyjski plan pokojowy omawiany przez przywódców arabskich w Bejrucie — Zamach palestyńskiego samobójcy w Netanii
17. W martwym punkcie — Izrael odrzucił saudyjski plan pokojowy — Palestyńczycy obawiali się izraelskiego odwetu
18. Misja Zinniego w cieniu zamachów — Palestyńskie akty terroru

**Przykład 4:** Najliczniejsza spośród grup znalezionych przez algorytm HAC-Threshold w korpusie K3

K3 znalazło się 8 grup z kolekcji K1, których zawartość się niemal w pełni zgadza z prawidłowym podziałem — jedynie dokument *Wyeliminowanie niewygodnego kandydata — bezkarne* nie został nigdzie przypisany.

Grupy skonstruowane przez algorytm DBSCAN dla progu sąsiedztwa równego 0.5 i minimalnej liczby sąsiadów równej 1 są bardzo podobne do tych utworzonych przez algorytm hierarchiczny — w 74% przypadków są takie same, w szczególności tak samo rozpoznanych zostało osiem klastrów z K1.

### Ocena etykiet dla grup

Osoby biorące udział w eksperymencie zostały poproszone również o ocenę jakości etykiet w skali 1 do 3, której znaczenie jest następujące

- 1 etykieta jest nieprawidłowa (22% etykiet grup otrzymało taką ocenę)
- 2 etykieta przybliży tematykę klastra, ale trudno na jej podstawie odgadnąć, jakiego typu artykuły się w nim znajdują (34% wskazań)
- 3 etykieta dobrze oddaje zawartość klastra (44% wskazań)

Średnia ocena wyniosła 2,21, grupy były etykietowane metodą 5 słów kluczowych. Należy przyznać, że jest to wynik dość dobry, wzięwszy pod uwagę prostotę konstrukcji etykiet.

---

# Podsumowanie

---

Celem pracy była ocena możliwości polepszenia wyników grupowania dokumentów tekstowych przez udoskonalanie sposobu ich reprezentacji. Eksperymenty pokazały, że jest to możliwe. Oprócz czynności standardowo wykonywanych dla języka polskiego, czyli redukcji wyrazów funkcyjnych oraz lematyzacji, warto przede wszystkim usuwać wszystkie leksemy poza rzeczownikami. Duże znaczenie ma również rozpoznawanie jedno- i wielowyrazowych nazw własnych i nadawanie im wyższych wag niż pozostałym terminom. Rozpoznawanie fraz miało w przeprowadzonych eksperymentach znaczenie niższe od oczekiwanego. Również wykorzystanie ontologii typu WordNet nie spełniło pokładanych w niej nadziei, lecz spowodowane jest to prawdopodobnie kontekstem badań — pożądanym było jak najściślejsze podobieństwo dokumentów, WordNet zaś umożliwia rozpoznanie bardziej subtelnych cech podobieństwa.

Porównane zostało działanie klasycznych algorytmów analizy skupień w grupowaniu dokumentów tekstowych. Najlepsze wyniki można było osiągnąć przy użyciu zmodyfikowanych algorytmów grupowania aglomeracyjnego UPGMA: HAC-Threshold i HAC-Flat. Nieco niższej jakości grupy zwracał bardzo szybki i prosty algorytm DBSCAN, który rzadko jest wykorzystywany do grupowania danych tekstowych. W przeprowadzonych testach algorytm k-średnich zwracał wyniki zawierające bardzo dużo błędów.

Podstawą kolejnych badań nad ulepszeniem grupowania dokumentów powinna być funkcja podobieństwa. Stosowana w aplikacji miara  $sim_{cosine}$  dobrze się sprawdza dla tradycyjnego modelu multizbioru słów ważonych funkcją  $w_{tf-idf}$  czy  $w_{tf}$ , lecz trudno przy jej użyciu właściwie oddać podobieństwo bazujące na bardzo silnych cechach wiążących dokumenty: frazach i nazwach własnych. Każde ich pojawienie się w obu porównywanych dokumentach wskazuje na zbliżoną tematykę. Zastosowany w aplikacji system współczynników ważenia dla terminów różnego typu nie zawsze poprawnie tę właściwość pozwala uwzględnić.

---

# Bibliografia

---

- [AF96] Nicholas O. Andrews and Edward A. Fox. Recent developments in document clustering. Technical report, Department of Computer Science, Virginia Tech, Blacksburg, 1996. [cytowanie na str. 21]
- [EpKSX96] Martin Ester, Hans peter Kriegel, Jörg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996. [cytowanie na str. 17]
- [For06] Richard Forster. *Document Clustering in Large German Corpora Using Natural Language Processing*. PhD thesis, University of Zurich, 2006. [cytowanie na str. 14, 28]
- [GJM09] Filip Graliński, Krzysztof Jassem, and Michał Marcińczuk. An environment for named entity recognition and translation. In *Proceedings of the 13th Annual Conference of the EAMT*, pages 88–96, Barcelona, marzec 2009. [cytowanie na str. 26]
- [HSS03] A. Hotho, S. Staab, and G. Stumme. Wordnet improves text document clustering. In Ying Ding, Keith van Rijsbergen, Iadh Ounis, and Joemon Jose, editors, *Proceedings of the Semantic Web Workshop of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval (SIGIR 2003), August 1, 2003, Toronto Canada*, 2003. Published Online at <http://de.scientificcommons.org/608322>. [cytowanie na str. 29]
- [Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967. [cytowanie na str. 11]
- [MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. 2008. [cytowanie na str. 2, 13, 16, 21, 49]
- [Por80] Martin F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980. [cytowanie na str. 25]
- [rze] Korpus rzeczpospolitej. [on-line] <http://www.cs.put.poznan.pl/dweiss/rzeczpospolita>. [cytowanie na str. 50]

- [SBLK05] Adam Schenker, Horst Bunke, Mark Last, and Abraham Kandel. Graph-theoretic techniques for web content mining. 2005. [cytowanie na str. 8, 9]
- [SKK00] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000. [cytowanie na str. 16, 17]
- [SWY75] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975. [cytowanie na str. 4]
- [Wei06] Dawid Weiss. *Descriptive Clustering as a Method for Exploring Text Collections*. PhD thesis, Poznań University of Technology, Poznań, Poland, 2006. [cytowanie na str. 21, 22]
- [ZE05] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. 2005. [cytowanie na str. 18]

# Dodatki



## Dodatek A

---

# Instrukcja użycia aplikacji Ugrupator

---

### A.1 Wymagania techniczne

Załączona aplikacja grupująca dokumenty wymaga do działania komputera z systemem operacyjnym MacOS w wersji co najmniej 10.5, Windows XP (lub późniejszy) lub Linux (z zainstalowanym środowiskiem uruchomieniowym Javy w wersji 6).

### A.2 Korzystanie z aplikacji

#### Przygotowanie dokumentów do grupowania

Aby dokumenty mogły zostać wczytane do reprezentacji standardowej, powinny one być przechowywane na dysku jako zwykłe pliki tekstowe w kodowaniu UTF-8.

Do reprezentacji rozszerzonej dokumenty muszą mieć format plików ITF. Format ten jest wewnętrznym sposobem przechowywania przez system tłumaczenia automatycznego Translatica przeanalizowanego leksykalnie i syntaktycznie tekstu. Za pomocą aplikacji *t5console* można przekształcić zwykły plik tekstowy *artykuł.txt* w plik w formacie ITF *artykuł.itf* w następujący sposób (polecenia konsoli bash):

```
cat artykuł.txt | ./t5console --lemmatize plen > artykuł.itf
```

Oba powyższe pliki (*artykuł.txt* i *artykuł.itf*) są potrzebne do grupowania i powinny znajdować się w tym samym katalogu.

### Sposób korzystania z aplikacji

1. wybrać *Narzędzia* → *konfiguracja*. W oknie, które się ukaże, należy ustawić pożądane opcje reprezentacji, tj. współczynniki ważenia dla poszczególnych części mowy, sposób ważenia terminów, decyzję o wykorzystaniu WordNetu i sposób tworzenia etykiet dla klastrów. Zapisać zmiany.
2. wybrać *Narzędzia* → [*wczytaj dokumenty itf* || *wczytaj dokumenty txt*] i zaznaczyć dokumenty do grupowania. Po akceptacji, na dole ekranu będą pojawiać się informacje o postępie we wczytywaniu plików.  
Uwaga: Nie ma możliwości dołączenia plików do stworzonej kolekcji, każdy wybór plików przez pozycję *Wczytaj dokumenty* tworzy nowy zbiór dokumentów do grupowania.
3. Aby zobaczyć reprezentację wczytanych dokumentów w postaci listy terminów i przypisanych im wag, należy wybrać *Narzędzia* → *Dokumenty*
4. Aby zobaczyć obliczone wartości podobieństwa między wszystkimi parami dokumentów, należy wybrać *Narzędzia* → *Macierz podobieństwa*. Kliknięcie na nagłówek kolumny powoduje sortowanie jej wartości.
5. Aby pogrupować dane, należy nacisnąć przycisk *Nowe grupowanie*, wybrać algorytm i wprowadzić wartości wymaganych parametrów. Po zakończeniu jego działania, wyniki będą prezentowane w 3 kolumnach: lista zidentyfikowanych grup, znajdujące się w zaznaczonej grupie dokumenty oraz podgląd zaznaczonego dokumentu.

## Dodatek B

---

# Korpus testowy

---

Tytuły dokumentów podzielone na grupy tematyczne zawarte w korpusie K1:

Nr	Dokumenty w grupie
1	<i>(bp.pl)</i> BP uaktualnia informacje na temat działań interwencyjnych w zatoce meksykańskiej <i>(rp.pl)</i> Pięć sposobów bp na plamę ropy <i>(wprost.pl)</i> Obama: BP zapłaci za katastrofę ekologiczną <i>(money.pl)</i> BP chce powstrzymać wyciek ropy nową kopułą <i>(gazeta.pl)</i> Przełom. BP udało się zamontować rurę wysysającą ropę <i>(gazeta.pl)</i> Wyciek ropy kosztował koncern 450 mln dol <i>(radiozet.pl)</i> Firma BP: Do wtorku chcemy zatamować wyciek ropy <i>(rp.pl)</i> Katastrofa platformy: plama ropy zagraża Ameryce
2	<i>(dziennik.pl)</i> Na PZU najlepiej zarobi jego prezes <i>(gazeta.pl)</i> Kurs PZU lekko w dół Bank UBS wycenia je na 395 zł <i>(rp.pl)</i> Kolejny dzień PZU
3	<i>(gazeta.pl)</i> Lech Kaczyński nie żyje. Katastrofa samolotu. 96 ofiar <i>(rp.pl)</i> Nie żyje prezydencka para, politycy, żołnierze <i>(tvn.pl)</i> Głowa państwa polskiego nie żyje
4	<i>(gazeta.pl)</i> Libia W katastrofie samolotu zginęły 103 osoby <i>(se.pl)</i> Katastrofa samolotu w Libii
5	<i>(gazeta.pl)</i> Ojciec Rydzyk uruchomił nowe radio <i>(onet.pl)</i> Ojciec Rydzyk uruchomił nowe radio <i>(wirtualnemedi.pl)</i> O Tadeusz Rydzyk uruchomił nowe radio
6	<i>(gazeta.pl)</i> Powódź 2010. można było zapobiec zalaniu Kozanowa <i>(polskieradio.pl)</i> Pogarsza się sytuacja na os. Kozanów <i>(se.pl)</i> Sytuacja na wrocławskim osiedlu Kozanów powoli się stabilizuje

7	<p><i>(gazeta.pl)</i> Sąd nie dał wiary wyjaśnieniom Jaruckiej</p> <p><i>(pap.pl)</i> Jarucka skazana na karę w zawieszeniu; obrona zamierza apelować</p> <p><i>(salon24.pl)</i> Wylimitowanie niewygodnego kandydata — bezkarne</p> <p><i>(tvp.pl)</i> Anna Jarucka skazana</p> <p><i>(wprost.pl)</i> Asystentka Cimoszewicza pójdzie do więzienia. Dziś wyrok</p>
8	<p><i>(dziennik.pl)</i> Kryzysowy festiwal w Gdyni</p> <p><i>(gazeta.pl)</i> Różyczka zwyciężcą festiwalu filmowego w Gdyni</p> <p><i>(emetro.pl)</i> Agentka, historia, Złote Lwy</p> <p><i>(newsweek.pl)</i> Różyczka najlepsza na festiwalu w Gdyni</p> <p><i>(pomorska.pl)</i> Różyczka laureatem Złotych Lwów na 35 Festiwalu Polskich Filmów Fabularnych w Gdyni</p> <p><i>(tvp.pl)</i> Różyczka zgarnęła wszystkie Złote Lwy</p>