Bazy
dokumentów tekstowych

Bazy dokumentów tekstowych

Dziedzina zastosowań

- Automatyzacja bibliotek
- Elektroniczne encyklopedie
- Bazy aktów prawnych i patentów
- Szukanie informacji w Internecie

Dokumenty tekstowe

- Książki – e-book
- Artykuły naukowe, gazetowe
- Dokumenty urzędowe
- Artykuły prawne
- E-maile
- Strony WWW

Specyfika

- Dane są niestrukturalne lub semistrukturalne
- Wyszukiwanie przybliżone (niejednoznaczne)
- Ranking znalezionych dokumentów

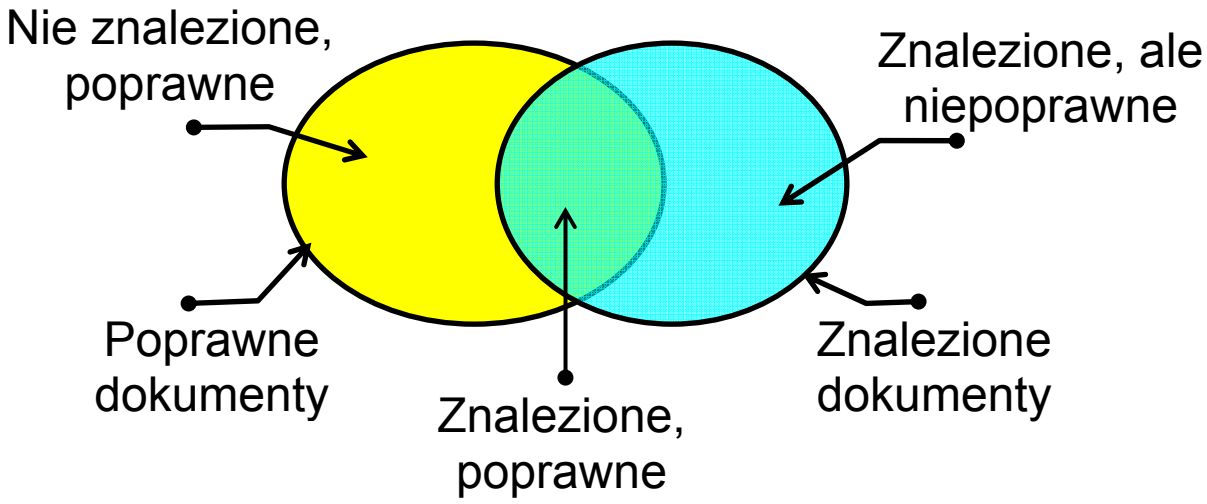
Pokrewne technologie informatyczne

- Wyszukiwanie informacji
- Wyszukiwanie tekstów
- Wyszukiwanie dokumentów
- Wyszukiwanie danych
- Wyszukiwarki Internetowe

Miary wyszukiwania dokumentów

W przeciwieństwie do systemów baz danych wyszukiwanie dokumentów zazwyczaj nie jest dokładne. Jakość wyszukiwania dokumentu jest zależna od jakości modelu reprezentacji dokumentów tekstowych, algorytmów wyszukiwania oraz od jakości zapytania. Spełnienie warunków wyszukiwania dokumentów tekstowych, nie jest binarne.

Wyniki wyszukiwania dokumentów:



Ustalenie *poprawności* znalezionych dokumentów należy do osoby wyszukującej. Termin poprawność oznacza jedynie potencjalną użyteczność dokumentów.

Miary jakości wyszukiwania

Powszechnie przyjęte jest stosowanie trzech miar jakości wyszukiwania w bazie dokumentów tekstowych.

Precyzja: jaki procent znalezionych dokumentów jest poprawnych

$$\textbf{Precision} = \frac{\left|\{poprawne\} \cap \{znalezione\}\right|}{\left|\{znalezione\}\right|}$$

Czułość: jaki procent poprawnych dokumentów zostało znalezionych

$$\textbf{Recall} = \frac{\left|\{poprawne\} \cap \{znalezione\}\right|}{\left|\{poprawne\}\right|}$$

Fall-out: jaki procent znalezionych dokumentów jest niepoprawny

$$\textbf{Fall-out} = \frac{\left|\{niepoprawne\} \cap \{znalezione\}\right|}{\left|\{znalezione\}\right|}$$

Fall-out = 1 - Precision

Metody wyszukiwania tekstów

Reprezentacja tekstu

- „goły” tekst – dane niestrukturalne
- metadane opisujące tekst: zbiór słów kluczowych, klasyfikacja tekstów, wyróżnione fragmenty tekstu (tytuły, śródtytuły, czcionka tekstu), wskazania na dokument tekstowy
- reprezentacja tekstów przez ich sygnatury
- reprezentacja tekstów jako punktów w przestrzeni wielowymiarowej

Klasy zapytań

- Szukanie za pomocą zapytania
  - wyrażenia logiczne (`and or not`) na słowach kluczowych
  - wagi przypisywane poszczególnym słowom
  - wyrażenia regularne

  Problemy: fleksja, liczba mnoga, synonimy, polisemia, brak naturalnej miary odległości (brak rankingu), słowa neutralne
- Szukanie za pomocą wzorcowych dokumentów tekstowych

Metody wyszukiwania

- Pełny przegląd tekstu
- Pliki odwrócone na tekście lub na opisie tekstu
- Pliki sygnaturowe
- NN w przestrzeni wielowymiarowej

Pełny przegląd tekstu

Wydajne metody wyszukiwania wzorcowego tekstu (w ogólności wyrażenia regularnego) w łańcuchu znaków:

- Algorytmy: Knutha-Morrisa-Pratta lub Boyera i Moora o liniowej złożoności obliczeniowej: **length(pattern) + length(string)**;
  Złożoność algorytmu podstawowego to:
  length(pattern) * length(string)

tekst: ALGORYTMY I STRUKTURY DANYCH
wzorzec: TUR

```
ALGORYTMY I STRUKTURY DANYCH
   | +3 | +3 | +3 | +3 | +3 |+2 |
TUR   |    |    |    |    |   |
    TUR   |    |    |    |   |
        TUR   |    |    |   |
            TUR   |    |   |
                TUR   |   |
                    TUR   |
                        TUR |
                          TUR
```

W danym kroku, przy braku dopasowania wyznaczane jest przesunięcie dla następnego kroku algorytmu.

Pełny przegląd tekstu

- Równoległe wyszukiwanie wielu słów;
- Tolerowanie błędów typowania słów wzorca poprzez wyszukiwanie wielu słów różniących się minimalnie od wzorca (ang. editing distance).

  Odległość Hamminga – określająca odległość między dwoma ciągami o tej samej długości, jako liczbę różniących się symboli.

  HD(mama, tata) = 2

  Odległość edycji dla dwóch tekstów jest minimalną liczbą operacji *insert* i *delete*, które pozwolą przekształcić jeden tekst w drugi.

  Dla dwóch tekstów x i y o długościach $|x|$ i $|y|$ odległość edycji wynosi:

  $$dE(x,y) = |x| + |y| - 2*|LCS(x,y)|$$

  gdzie LCS(x,y) jest najdłuższym wspólnym łańcuchem symboli zawartym w tekstach x i y.

  Na przykład, niech x='abcdefg', y='bcefh'. Wtedy: LCS(x,y) = ‘bcef’.

  dE(x,y) = 7 + 5 – 2 * 4 = 4

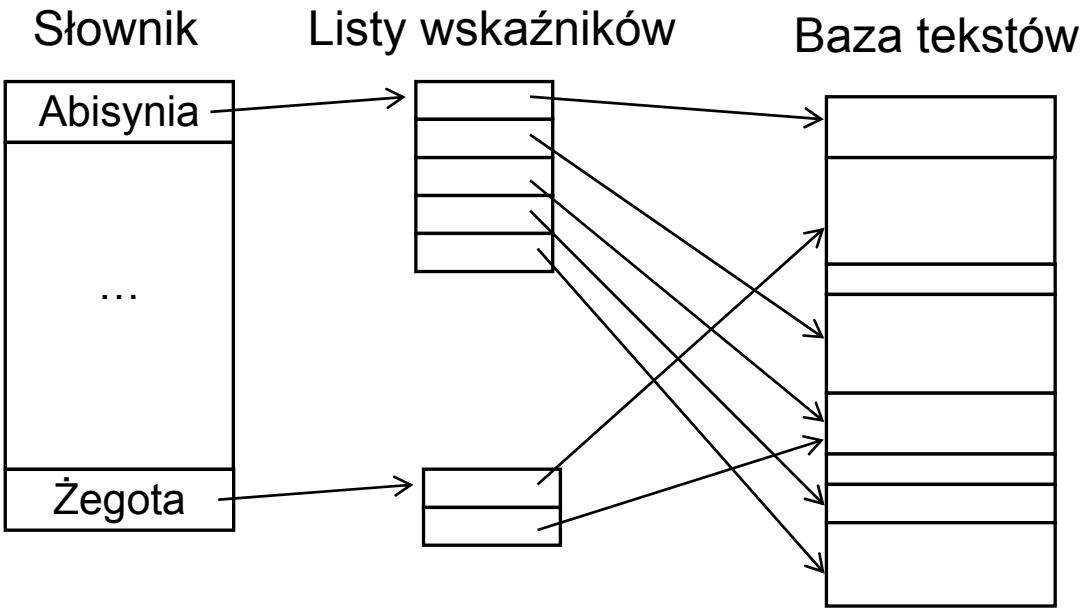
Odległość Levenshteina - dla dwóch tekstów jest minimalna liczbą operacji *insert*, *delete* i *update* które pozwolą przekształcić jeden tekst w drugi.

Własności metody

- brak narzutu dodatkowych struktur danych
- brak narzutu dla operacji modyfikacji tekstów
- mała wydajność dla dużych zbiorów dokumentów

Pliki odwrócone

Pliki/indeksy odwrócone (ang. inverted files/indexes) są specyficznym rodzajem indeksów pomocniczych (ang. secondary indices) zakładanych na poszczególnych słowach tekstów.



Różne rozszerzenia i wariacje:

- liczniki częstości występowania słów;
- pomijanie słów pospolitych;
- listy synonimów w słowniku;
- uproszczony indeks (ang. coarse index) dwupoziomowy Glimps adresuje zbiory tekstów (mniej i krótsze wskaźniki); rozmiar indeksu jest równy 2-4% wielkości bazy danych tekstów

Własności metody

- bardzo wydajne wyszukiwanie
- duży narzut na dodatkowe struktury danych (50-300%)
- duży narzut na operacje modyfikacji

Typy plików odwróconych

Typy plików odwróconych

Ze względu na typ używanych wskaźników

- **nie-pozycyjne** – wskazania na dokumenty, jeden wskaźnik na cały dokument zawierający dane słowo
- **częstościowy** - wskazania na dokumenty poszerzone o liczbę wystąpień słowa, jeden wskaźnik na cały dokument zawierający dane słowo
- **pozycyjne** – wskazania na wystąpienia słowa w tekście, wiele wskaźników na różne pozycje tego samego słowa

Ze względu na sposób konstrukcji słownika

- **słownikowe** – predefiniowany słownik dla bazy dokumentów tekstowych
- **nie-słownikowe** – słowa występujące w indeksowanych tekstach

Przykładowa charakterystyka pliku odwróconego (5GB)

Table 5.4 Typical sizes and performance figures.

Parameter	Symbol	Assumed value
Total text size	B	5×10^9 bytes
Number of documents	N	5×10^6
Number of distinct words	n	1×10^6
Total number of words	F	800×10^6
Number of index pointers	f	400×10^6
Final size of compressed inverted file	I	400×10^6 bytes
Size of dynamic lexicon structure	L	30×10^6 bytes
Disk seek time	t_s	10×10^{-3} sec
Disk transfer time per byte	t_r	0.5×10^{-6} sec
Inverted file coding time per byte	t_d	5×10^{-6} sec
Time to compare and swap 10-byte records	t_c	10^{-6} sec
Time to parse, stem, and look up one term	t_p	20×10^{-6} sec
Amount of main memory available	M	40×10^6 bytes

Sygnatury tekstów

Sygnatury są uproszczoną numeryczną reprezentacją tekstów. Dzięki temu zajmują mniej miejsca i koszt ich przeszukiwania jest znacznie mniejszy, niż pełny przegląd bazy danych tekstów. Wartość sygnatury jest zależna od słów występujących w tekście, natomiast nie jest zależna od kolejności i częstości ich występowania. Sygnatura tekstu jest generowana na podstawie sygnatur wszystkich różnych słów występujących w tekście.

Dokładność odwzorowanie tekstu przez sygnaturę jest ograniczona. Różne teksty, zawierające różne słowa, mogą mieć takie same sygnatury. Pliki sygnatur reprezentujące zbiory tekstów są wykorzystywane jako filtr ograniczający rozmiar przeszukiwanego zbioru tekstów.

Sposoby generowania sygnatur tekstu

- Konkatenacja sygnatur

Sygnatura dokumentu jest tworzona jako sklejenie sygnatur wszystkich różnych słów występujących w dokumencie.

dokument	Ala	ma	małego	kota
	↓	↓	↓	↓
sygnatury słów:	0000	0001	0010	0011
sygnatura dokumentu:	0000 0001 0010 0011			

Szukanie dokumentów zawierających określone słowo polega na znalezieniu w pliku sygnatur wszystkich dokumentów, których sygnatura zawiera sygnaturę szukanego słowa.

Sygnatury tekstów

Sposoby generowania sygnatur

- Kodowanie nałożone (ang. superimposed coding)

  Każdy tekst dzielony jest na logiczne bloki o tej samej liczbie różnych niepospolitych słów. Sygnatury słów mają stałą długość `F`. Każde słowo zajmuje w sygnaturze `m` bitów. Ustawione bity różnych słów mogą się pokrywać, unikalna jedynie jest ich kompozycja. Sygnatura bloków danego tekstu jest tworzona przez zastosowanie operacji `OR` na sygnaturach wszystkich słów w bloku. Sygnatura dokumentów jest tworzona przez konkatenację sygnatur bloków dokumentu.

słowo	sygnatura
„Ala”	0000 0010 0000 0001 1000
„ma”	0001 0010 0100 0000 0000
„kota”	1110 0000 0000 0000 0000
blok	1111 0010 0100 0001 1000
„Ala”	0000 0010 0000 0001 1000

Szukanie polega na porównywaniu sygnatur podanych słów z sygnaturami bloków dokumentów. Znalezione dokumenty są w ogólności nadzbiorem szukanego zbioru. Zwrócony zbiór dokumentów może zawierać również niewłaściwe dokumenty (ang. false drops).

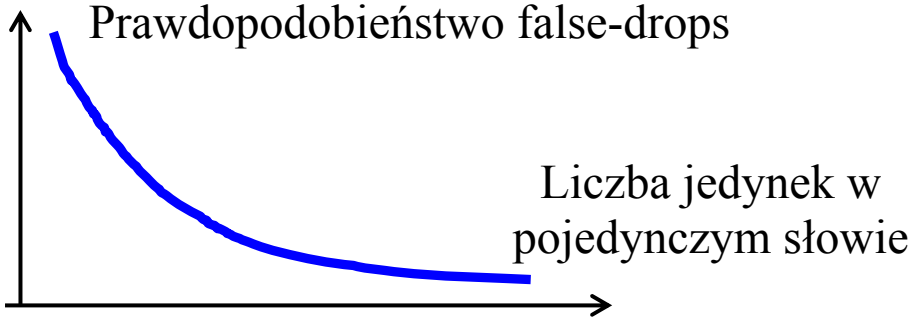
Optymalizacja parametrów plików sygnatur

Jakość szukania za pomocą plików sygnatur jest określona przez skuteczność ograniczenia rozmiaru przeszukiwanego zbioru dokumentów. Można ją oszacować jako proporcję rozmiaru szukanego zbioru dokumentów do rozmiaru znalezionego zbioru. Jakość szukania jest zależna od przyjętych parametrów sygnatur.

Koszt szukania:

$$\text{Koszt} = \text{Funkcja}(F, m, D_{blok})$$

gdzie:
F - rozmiar sygnatur słów;
m - liczba jedynek w pojedynczym słowie;
D_{blok} – rozmiar bloków mierzony liczbą różnych słów.



Własności metody

- wydajne wyszukiwanie
- mały narzut na dodatkowe struktury danych (~10%)
- mały narzut na operacje modyfikacji
- wymaga umiejętnego zaprojektowania

Wektorowa reprezentacja tekstów

Dokumenty są reprezentowane jako N-wymiarowe wektory, gdzie N jest liczbą wszystkich pojęć (słów kluczowych) znajdujących się w bazie dokumentów. Wektory mogą być mapami bitowymi; 1 oznacza występowanie, a 0 brak danego pojęcia w dokumencie; lub wektorami liczbowymi, gdzie poszczególne wartości reprezentują liczbę wystąpień danego słowa w dokumencie.

Przykład:

Dana jest przestrzeń wielowymiarowa

MD = <algorytmy, analizy , metody, grupowanie>

Tekst:

> Algorytmy analizy skupień dzieli się na szereg kategorii: Większość metod analizy skupień to metody iteracyjne, wśród których najczęściej używane to metody kombinatoryczne: aglomeracyjne i deglomeracyjne.

jest reprezentowany przez wektor logiczny:

<1, 1, 1, 0 >

lub wektor liczbowy:

<1, 2, 2, 0 >

Własności reprezentacji wektorowej

- Nie oddaje kolejności występowania poszczególnych słów w tekście
- Dla dużej liczby wymiarów wektory reprezentujące teksty są bardzo rzadkie, tzn. większość składowych wektorów ma wartość równą zero.
- Pozwala na wyszukiwanie za pomocą zapytania i za pomocą wzorcowych tekstów. Zapytania muszą być reprezentowane za pomocą wektorów.
- Reprezentacja wektorowa charakteryzuje się naturalną miarą odległości między tekstami. Pozwala to na ustalenie rankingu znalezionych tekstów oraz na wyszukiwanie przybliżone.

Miary odległości tekstów

Odległość kosinusowa

Poszczególne teksty są punktami w przestrzeni wielowymiarowej określonej przez słowa występujące w tekstach. Kąt między wektorami tekstów reprezentuje ich podobieństwo.

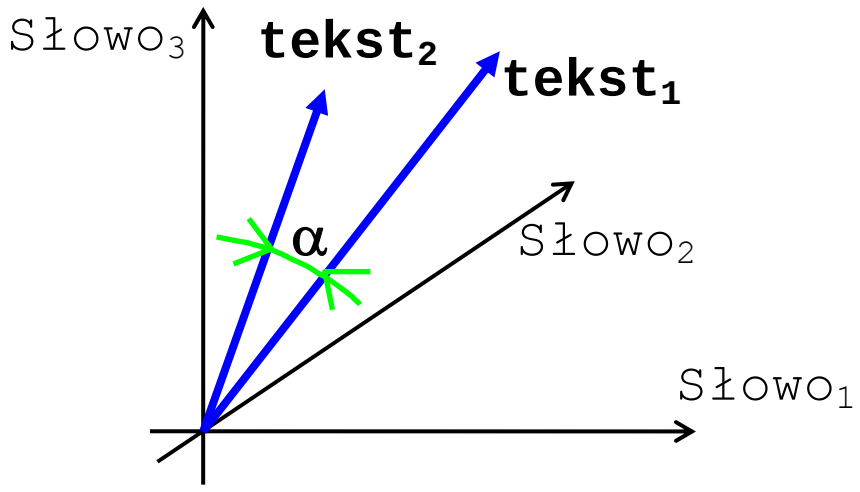
$$\cos(\mathrm{v1}, \mathrm{v2}) = \frac{v_1 * v_2}{|v_1| * |v_2|}$$

Jeżeli kąt między wektorem zapytania i wektorem danego dokumentu lub między wektorami dwóch dokumentów są równe zero, to odległość kosinusowa jest równa 1.

Zapytanie o teksty o *algorytmach grupowania* przetransformowane do postaci wektorowej:

Q = <1,0,0,1>

Wyszukiwane są teksty, o najmniejszej odległości kosinusowej od wektora zapytania. Teksty mogą zostać uporządkowane ze względu na zmniejszającą się odległość kosinusową.



Wagi słów

„Baza danych **jest dużą** kolekcją danych.”

„Ania **jest dużą** dziewczynką.”

Które współdzielone słowa są istotne dla podobieństwa tekstów?

- Waga TF (term frequency) – reprezentuje częstość występowania słowa w tekście. Waga TF preferuje długie teksty i nie oddaje siły dyskryminacji poszczególnych słów.
- Waga IDF (inverse document frequency) - reprezentuje siłę dyskryminacji tekstów przez dane słowo.

  $$IDF = \log (N/n_j)$$

  gdzie: N jest liczbą wszystkich tekstów, a n_j liczbą tekstów, w których występuje dane słowo.

Waga słowa w wektorze reprezentującym dany tekst jest iloczynem wag TF i IDF.

Baza danych jest dużą kolekcją danych

<Ania, baza, dana, duża, dziewczyna, jest, kolekcja>

<0*0.3, 1*0.3, 2*0.3, 1*0, 0*0.3, 1*0, 1*0.3>

<0, 0.3, 0.6, 0, 0, 0, 0.3>