Uniwersytet Kardynała Stefana Wyszyńskiego
w Warszawie
Wydział Matematyczno-Przyrodniczy
Szkoła Nauk Ścisłych

Jacek Giedronowicz
Nr albumu: 95175

Rekomendacja stron www z silnikiem Apache Lucene

Praca licencjacka na kierunku *Informatyka*
w zakresie *Machine Learning*

Praca wykonana pod kierunkiem
dr. Roberta Kłopotka

Lipiec 2020

Słowa kluczowe

aplkacja webowa, wyszukiwarka, Apache Lucene

Dziedzina Socrates-Erasmus

11.3 Informatyka

Klasyfikacja tematyczna

Machine Learning

English title

Website recommendation using the Apache Lucene engine

Spis treści

Rozdział 1

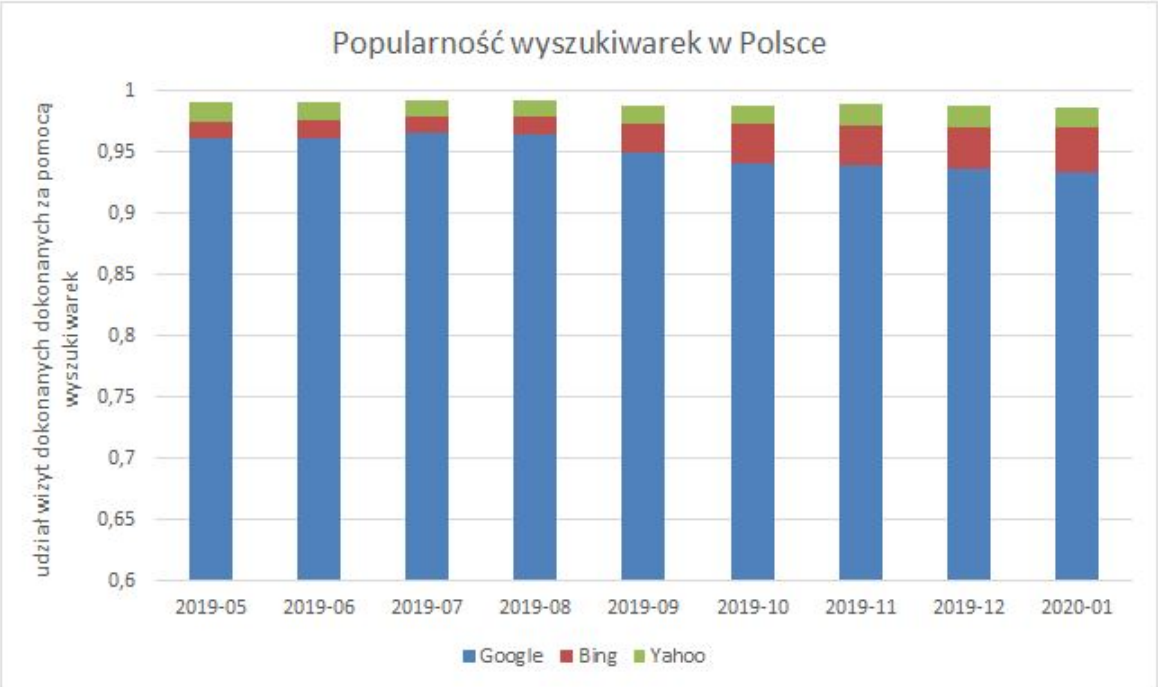
Wprowadzenie

Rozdział 2

Stan rynku

Na rynku istnieje wiele wyszukiwarek internetowych, lecz mało kto potrafi wymienić więcej niż pięć. Internet zdominowała firma Google. Nawet takie znane wyszukiwarki jak Bing od Microsoft'u czy Yahoo mają zaledwie kilka procent udziału, który przedstawia się następująco:

- Google - 93,37%
- Bing - 3,61%
- Yahoo - 1,75%
- DuckDuckGo - 0,29%
- Interia Katalog - 0,14%



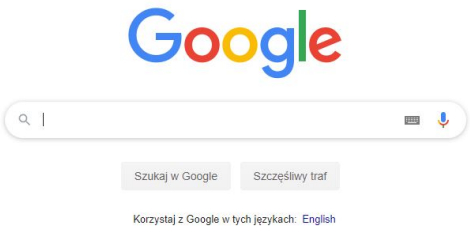
Rysunek 2.1: Popularność wyszukiwarek w Polsce
Źródło: gs.statcounter.com – styczeń 2020

W Polsce, z powodu trudności w przetwarzaniu naszego języka przez zagraniczne systemy, były próby stworzenia wyszukiwarek dedykowane dla polaków. Systemy z algorytmami rozpoznawania odmian słów w języku polskim. Największe powodzenie miała witryna szukacz.pl, która funkcjonowała przez 10 lat od 2001 do 2011 roku. Jak możemy przeczytać na stronie http://szukacz.pl/wiecej_o_szukaczu.html [7]:

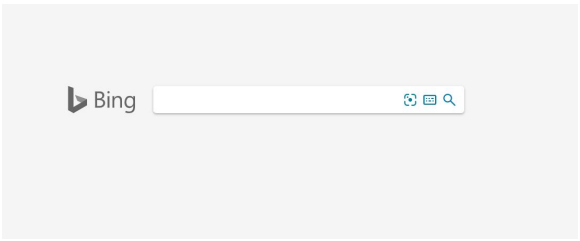
> "W szczycie swojego rozwoju, pod koniec 2007 roku, odpowiadał ze 115 milionów dokumentów w języku polskim pochodzących z miliona witryn (kolekcja „Polska"). Do połowy 2007 roku odpowiadał także z 45 milionów wyselekcjonowanych dokumentów w języku angielskim pochodzących z 2 milionów witryn (kolekcja „Świat"; tylko 4 procent pytań było skierowane do tej kolekcji)."

Mimo sporej bazy dokumentów oraz przyzwoitym budżecie, również i ta witryna musiała się poddać międzynarodowemu gigantowi jakim jest Google. Stąd też wybrałem witrynę google.pl jako wzór, do którego będę się odnosił przy implementacji własnego systemu. W dalszej części opiszę najważniejsze funkcjonalności witryny Google i porównam z Bing.

2.1. Wyszukiwarka

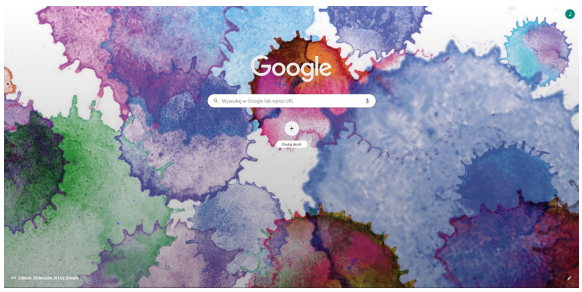


Rysunek 2.2: Główna strona wyszukiwarki google

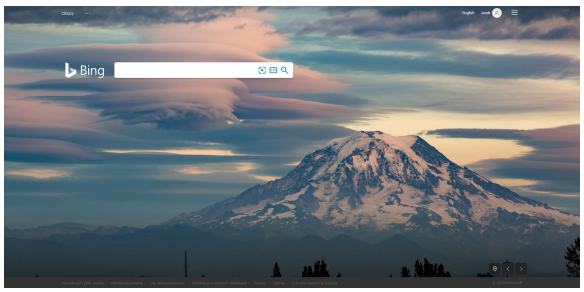


Rysunek 2.3: Główna strona wyszukiwarki bing

Na rysunkach 2.2 i 2.3 przedstawiono główne strony wyszukiwarek Google i bing. Jak widać obie strony są minimalistyczne i intuicyjne. Witryny oferują również możliwość jej spersonalizowania przez bogatą bazę skórek, co zostało przedstawione na rysunkach 2.4 i 2.5



Rysunek 2.4: Główna strona wyszukiwarki Google ze skórką

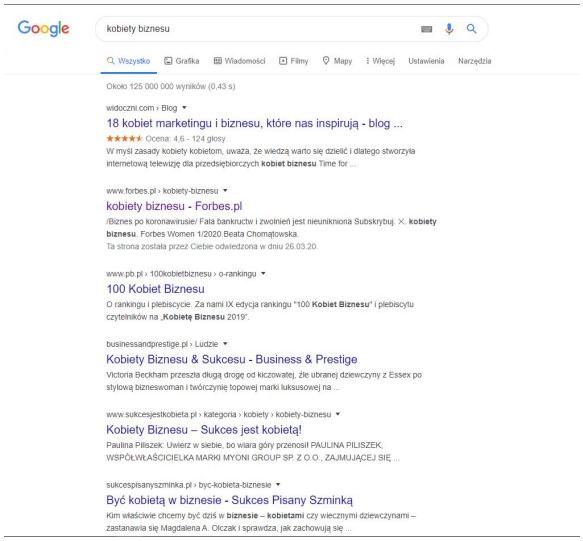


Rysunek 2.5: Główna strona wyszukiwarki Bing ze skórką

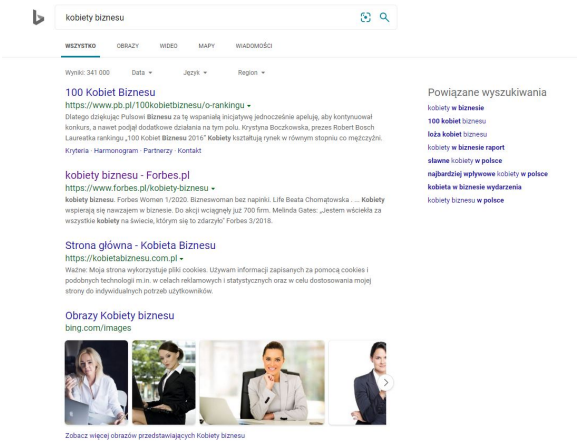
2.2. Wyszukiwanie

Po wpisaniu frazy, którą chcemy wyszukać obie wyszukiwarki wypiszą nam listę od kilkudziesięciu tysięcy do nawet kilkuset milionów rekomendowanych stron dla tego hasła.

Dla przykładu dla frazy „kobiety biznesu" Google znalazło 125 000 000 wyników kiedy Bing zaledwie 341 000. Dominacja wyszukiwarki Google wśród użytkowników oraz zdecydowanie większa pula zaindeksowanych stron może być powodem dlaczego współcześni deweloperzy przygotowują swoje strony głównie pod pozycjonowanie wyszukiwarki Google.



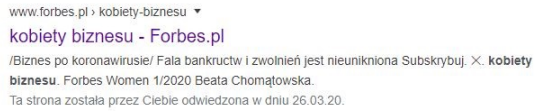
Rysunek 2.6: Lista rekomendowanych przez Google stron dla hasła: kobiety biznesu



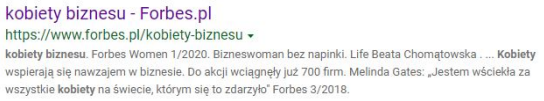
Rysunek 2.7: Lista rekomendowanych przez Bing stron dla hasła: kobiety biznesu

Każda z wyświetlanych pozycji przekazuje nam podstawowe informacje takie jak:

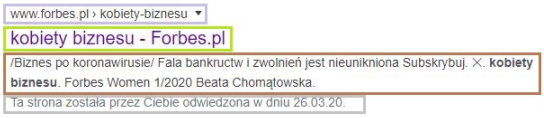
- tytuł strony (zaznaczony kolorem zielonym)
- link do strony (zaznaczony kolorem fioletowym)
- skrót zawartości strony (zaznaczony kolorem brązowym)



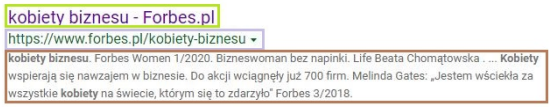
Rysunek 2.8: Wyświetlana pozycja w liście wyników google



Rysunek 2.9: Wyświetlana pozycja w liście wyników bing



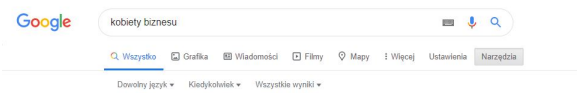
Rysunek 2.10: Wyświetlana pozycja w liście wyników Google z zaznaczonymi obszarami



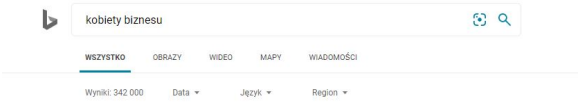
Rysunek 2.11: Wyświetlana pozycja w liście wyników Bing z zaznaczonymi obszarami

Oprócz podstawowego wyszukiwania linków do stron, mamy możliwość wybrania kategorii przedstawione na rysunkach 2.12 i 2.13 takich jak:

- grafika
- wiadomości
- filmy



Rysunek 2.12: Kategorie i filtry wyszukiwania Google



Rysunek 2.13: Kategorie i filtry wyszukiwania Bing

Ponadto wyszukiwarki dają nam możliwość intuicyjnego filtrowania wyników. W tym punkcie wyszukiwarki nieznacznie się różnią, gdyż Bing oferuje takie parametry jak: data, język, region. Natomiast Google: data, język i możliwość pokazania dokładnych wyników zawężając o wyniki fraz bliskoznacznych.

2.3. Podstawowe operatory

Aby podnieść jakość i zawęzić pole poszukiwań pożądanych informacji przez użytkownika, Google zaimplementowało tak zwane operatory. Operatory definiują zakres poszukiwań poprzez słowa i znaki kluczowe:

- *" "* - wpisując wyrażenie w cudzysłów wymusza ścisłe dopasowanie. Przykład: fraza *"Steave Jobs"* wpisana z cudzysłowem wymusza aby w wyniku były dokładnie te dwa słowa obok siebie.

- *OR* - operator logiczny pozwalający z zapytanie *Steave OR Jobs* wyszukać wyniki zawierające frazę *Steave* lub *Jobs* lub oba. Operator można zastąpić znakiem |.
- *AND* - domyślny operator logiczny, wyświetlający tylko te wyniki, które zawierają frazę *Steave* i *Jobs*. Operator jest domyślny więc przydatny jest tylko w połączeniu z innymi operatorami.
- *−* - operator wykluczający. W przykładzie *jaguar speed -car* wynikiem będą strony powiązane z frazą *jaguar speed* ale niezawierające *car*. Zatem chodziło nam o znalezienie prędkości jaguara będącego zwierzęciem a nie samochodem.
- *..* - operator zakresu liczb. Przykład: *podatki 2010..1015* da nam wynik wyszukiwania związanymi z podatkami miedzy 2010 a 2015 rokiem.
- *( )* - operator grupowania fraz z innymi operatorami. Przykład: *(iPad OR iPhone)apple*. Operator *AND* jest domyślny więc nie jest obowiązkowy.
- *define:* - operator wyszukuje definicje szukanej frazy.
- *filetype:* lub *ext:* - operator określający rodzaj pliku będący wynikiem wyszukiwania takie, jak pdf, docx, txt, ppt. Przykład: *Steve Jobs ext:pdf* - wyświetli tylko pliki pdf związane ze Steve'em Jobs'em.
- *site:* - ogranicza wyniki wyszukiwania tylko do jednej strony lub domeny. Przykład: *wniosek site:.gov.pl* wyszuka informacje o wnioskach tylko na stronach rządowych kończących się na *.gov.pl*.
- *inurl:* - operator przeszukuje podaną frazę tylko w adresie url strony.
- *intitle:* - operator przeszukuje podaną frazę tylko w tytule strony.
- *intext:* - operator przeszukuje podaną frazę tylko w treści strony. Pominie strony zawierające frazę w tytule czy adresie url.

Google zawiera znacznie więcej operatorów. Niektóre z nich wymienione są na stronie sites.google.com [4]

Rozdział 3

Rekomendacja stron www

Cały algorytm możemy podzielić na dwa etapy: indeksowanie i wyszukiwanie. Użytkownik oczekuje od aplikacji szybkiego i trafnego wyszukiwania jednak, aby tak się stało trzeba najpierw zebrać wszystkie dane i nadać im specjalną strukturę dzięki której będziemy mogli szybko znaleźć interesującą nas informacje. To właśnie nazywamy procesem indeksowania.

3.1. Proces indeksowania

3.1.1. Document

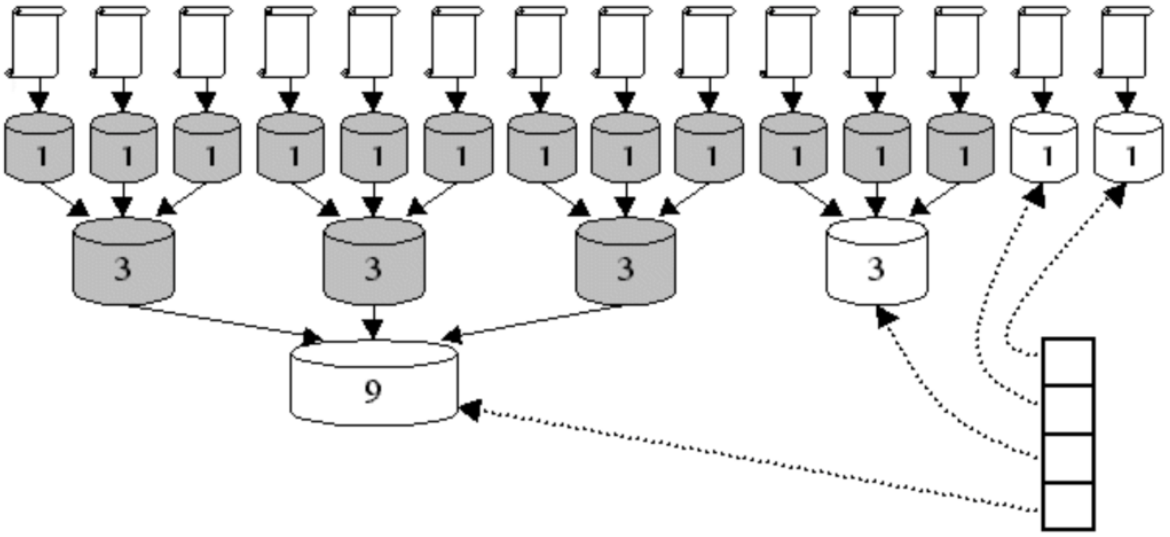
Dokument reprezentuje stronę internetową. Jest główną jednostką indeksowania i wyszukiwania. Technicznie jest zbiorem (kontener typu `Set`) zawierający pola (`field`) złożone z klucza i wartości. Dzięki temu możemy zapisać w ustrukturyzowany sposób najważniejsze dla nas informacje, po których później będziemy wyszukiwać. Do wyszukiwania stron www przydatne będą takie pola jak :

Rysunek 3.1: Struktura dokumentu

- `id` - unikalny identyfikator dokumentu
- `url` - adres url strony
- `path` - ścieżka do kodu html na dysku twardym
- `title` - tytuł strony
- `keywords` - słowa kluczowe z meta znacznika `<meta name="keywords">`
- `description` - opis strony z mata znacznika`<meta name="description">`
- `headers` - nagłówki ze znaczników `<h1>` do `<h6>`
- `emphasize` - wyróżnione słowa za pomocą znaczników takich jak `<em>,<b>,<strong><i>,<mark>`
- `images` - informacje o obrazach dostępnych na stronie

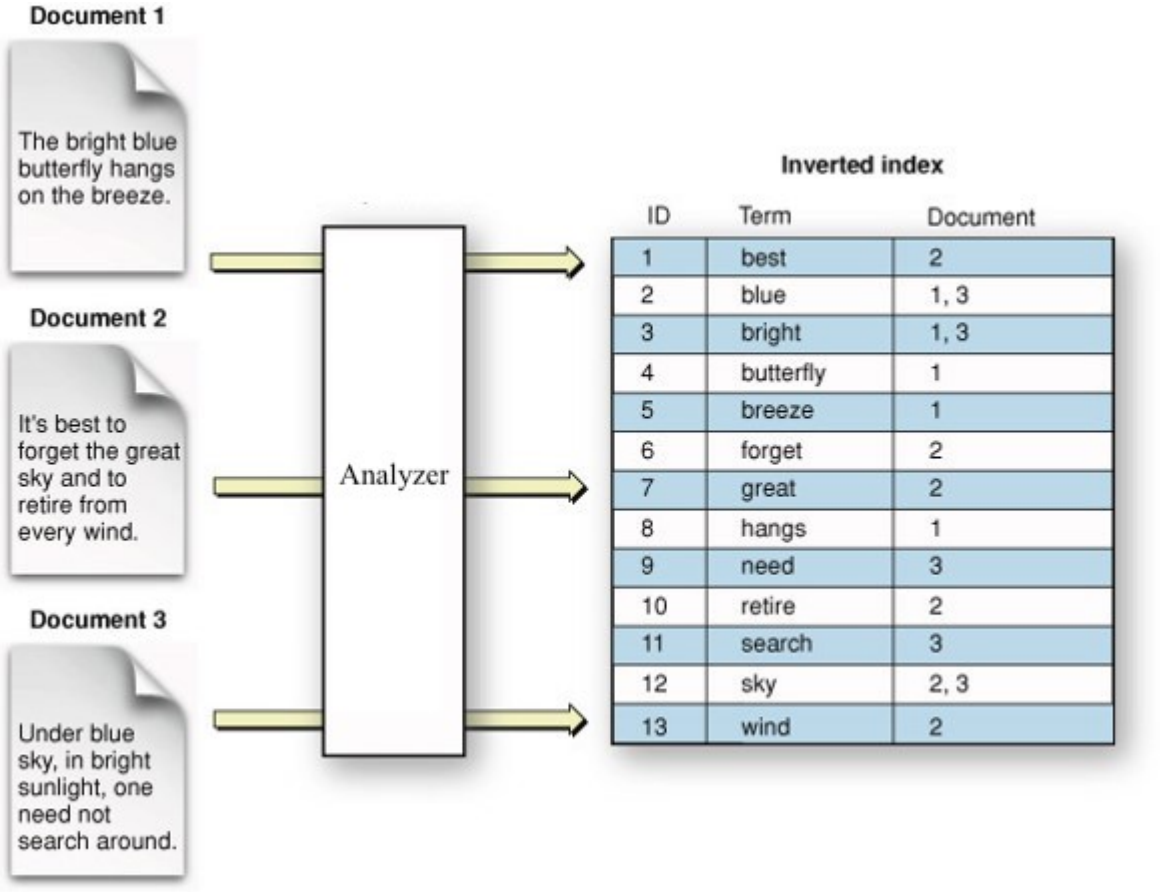
3.1.2. Struktura indeksu

Index jest pojedynczym katalogiem na dysku struktury drzewiastej [6]. Składa się z segmentów (`segment`), dokumentów (`document`), pól (`field`), wyrażeń (`term`). Segmenty można traktować jak pod-indeks jednakże nie stanowią one niezależnych indeksów. Liczba tworzących je plików zależy od liczby pól zawartych w indeksie. Wszystkie pliki należące do tego samego segmentu mają taki sam prefiks ale różnią się sufiksem.



Rysunek 3.2: Struktura indeksu [1]

Wyrażenia zapisane w dokumencie (`term`) są analizowane i zapisywane w specjalnej strukturze zwanej inverted index. Polega ona na przedstawieniu relacji `document - term` w postaci tabeli przedstawionej na rysunku 3.3. Możemy nie niej szybko znaleźć wyrażenie i id dokumentu, który zawiera dane wyrażenie.



Rysunek 3.3: Struktura indeksu

Powyższy sposób pozwala nam na szybkie znalezienie interesującego nas wyrażenia - dzięki analyzerowi wyrażenia będące uosobieniem tego samego słowa, niezależnie od formy gramatycznej, zajmują jeden rekord w bazie w postaci najprostszego wyrażenia.

3.1.3. Analyzer

Analyzer jest mechanizmem analizowania tekstu i generowania tak zwanych `token`'ów - będącymi jednostkami leksykalnymi. Dzięki temu dla programu słowa "wyrazić, wyrażenia, wyrażenie" to jedno i to samo.

Dla różnych potrzeb i co ważniejsze dla różnych języków zostały stworzone różne analyzery. Przykładowo:

- `WhitespaceAnalyzer`
  Rozdziela tekst bazując na białych znakach (spacja). Przykład.

```
["This is an example term analysis test."]

↓

["This", "is", "an", "example", "term", "analysis", "test."]
```

- `SimpleAnalyzer`
  Rozdziela tekst bazując na nieliterowych znakach takich jak liczby, znaki białe czy interpunkcja i zmienia je na małe znaki. Przykład.

  `["This is an 1example term analysis test."]`

  ↓

  `["this", "is", "an", "example", "term", "analysis", "test"]`

- `StopAnalyzer`
  Rozdziela tekst bazując na białych znakach. Usuwa interpunkcje oraz takie słowa jak 'a', 'an', 'the' itp Przykład.

  `["This is an example term analysis test."]`

  ↓

  `["example", "term", "analysis", "test"]`

- `StandardAnalyzer`
  Standardowy analyzer obsługujący nazwy, adresy email itp. Zmienia wszystkie znaki na małe oraz usuwa interpunkcje i zaimki/przedimki. Przykład.

  `["This is an example term analysis test."]`

  ↓

  `["example", "term", "analysis", "test"]`

- `PolishAnalyzer`
  Odpowiedni analyzer dla języka polskiego. Zmienia litery na małe, usuwa interpunkcje i co najważniejsze zmienia podane wyrażenia na ich podstawową formę (o ile to możliwe bezokolicznik lub mianownika). Przykład.

  `["To jest przykładowy test analizy wyrażenia."]`

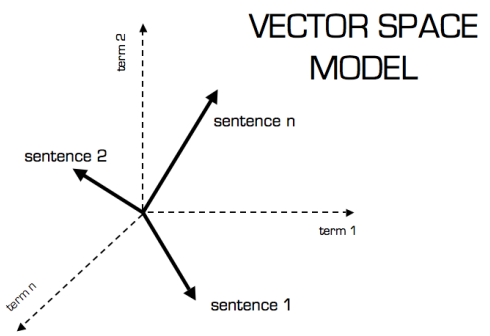
  ↓

  `["przykładowy", "test", "analiza", "wyrazić"]`

W procesie wyszukiwania będziemy musieli użyć dokładnie tego samego analyzera do przygotowania zapytania.

3.2. Proces wyszukiwania

Po zaindeksowaniu wszystkich dokumentów możemy je teraz przeszukiwać. Proces wyszukiwania przechodzi po indeksie szukając najlepszego dopasowania do naszego zapytania (`Query`), przeanalizowanego przez nasz analyzer. Do punktowania (`Scoring`) wykorzystywany jest model podobieństwa TF-IDF (`TF-IDF Similarity`) implementujący model przestrzeni wektorowej (`Vector Space Model - VSM`). W `VSM` im więcej razy dane słowo pojawia się w dokumencie w stosunku do frekwencji we wszystkich dokumentach tym wyżej punktowany jest.



Rysunek 3.4: VSM

3.2.1. TF-IDF Similarity

Podobieństwo TF-IDF (term frequency – inverse document frequency) [2] jest metodą obliczania wagi słów w oparciu o liczbę ich wystąpień. Mierzone jest w modelu VSM, w którym zarówno dokumenty jak i zapytania reprezentowane są jako wektory ważone w przestrzeni wielowymiarowej.

Załóżmy, że t oznacza term oraz d dokument. Wartość podobieństwa TF-IDF termu t dla dokumentem d oblicza się ze wzoru:

$$tfidf(t) = tf(t,d) \times idf(t)$$

gdzie $tf(t,d)$ nazywamy frekwencją termu (`term frequency`) i wyrażamy wzorem

$$tf(t,d) = \frac{n_{t,d}}{\sum_k n_{k,d}}$$

gdzie $n_{t,d}$ jest liczbą wystąpień termu t w dokumencie d.

$idf(t)$ nazywamy odwróconą frekwencją dokumentów (`inverse document frequency`) wyrażona wzorem

$$idf(t) = \log \frac{|D|}{1 + |\{d\colon t \in d\}|}$$

gdzie $|D|$ jest liczbą dokumentów a $|\{d\colon t \in d\}|$ jest liczbą dokumentów zawierających dany term. Dodając jedynkę tj $1 + |\{d\colon t \in d\}|$ zabezpieczamy się przed dzieleniem przez 0.

Dla lepszej jakości wyszukiwań i bezpieczeństwa, dobra wyszukiwarka nie polega tylko na podobieństwie. Bierze również pod uwagę czy strona znajduje się w cenionej witrynie, czy witryna jest niskiej jakości albo nawet spamuje. Takie witryny powinny być odnotowane na specjalnej liście spamerskiej a wyszukiwarka nie powinna rekomendować żadnych stron z danej witryny.

Za to punktacje wysokiej jakości stron może podnieść tak zwany PageRank.

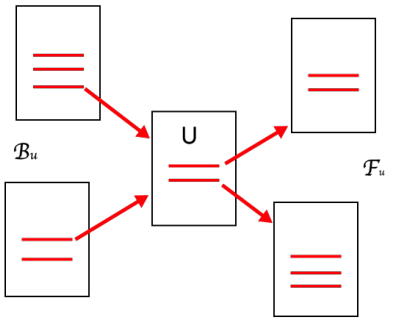
3.2.2. PageRank

Algorytm opracowany przez założyciela Google - Larry'ego Page'a i Sergey'a Brin'a [3]. Kluczową ideą jest rozważanie hiperłączy z jednej strony internetowej na drugą. Im więcej linków prowadzi do danej strony, tym ważniejsze ma znaczenie. Innymi słowy, jeśli strona internetowa jest wskazywana przez inne, ważne strony, to jest to również ważna strona.

Sieć połączeń stron hiperłączami tworzy graf skierowany.

Definicja 1. $G := (V, E)$ nazywamy grafem skierowanym. Zbiór $V \neq \emptyset$ jest zbiorem elementów zwanych wierzchołkami. Zbiór $E \subseteq \big\{\{u, v\} \colon u, v \in V\big\}$ nazywamy zbiorem krawędzi skierowanych z wierzchołka $u \in V$ do wierzchołka $v \in V$ oraz $\deg(u)$ liczbą krawędzi wychodzących z wierzchołka $u \in V$.

Definicja 2. Niech u będzie stroną internetową. Wtedy F_u będzie zbiorem stron, na które u wskazuje oraz B_u zbiorem stron, które wskazują na u.

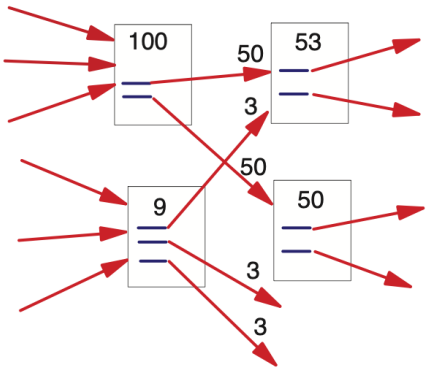


Rysunek 3.5: Zbiory F_u i B_u

PageRank R strony u jest zdefiniowany następująco:

$$R(u) = \frac{1-d}{N} + d \cdot \sum_{i=1}^{|B_u|} \frac{R(B_i)}{\deg(B_i)}$$

gdzie N jest liczbą dokumentów oraz d jest parametrem wybranym eksperymentalnie (zwykle $d = 0.85$) nazywany współczynnikiem tłumienia (*Damping factor*). Suma wartości wektora $R = \big[R_{u_1}, R_{u_2}, \ldots R_{u_N}\big]$ jest równa 1, tj. $\sum_{i=0}^{N} R_{u_i} = 1$.



Rysunek 3.6: Przykład uproszczonego PageRank

Powstaje problem typu *Cold Start* - do obliczeń PageRank danej strony potrzebujemy PageRanku stron do niej podlinkowanych, gdzie początkowo żadna ze stron nie posiada wyliczonego rankingu.

Wartości te można wyliczyć w sposób iteracyjny. Przyjmijmy początkowy rozkład prawdopodobieństwa dla czasu $t = 0$:

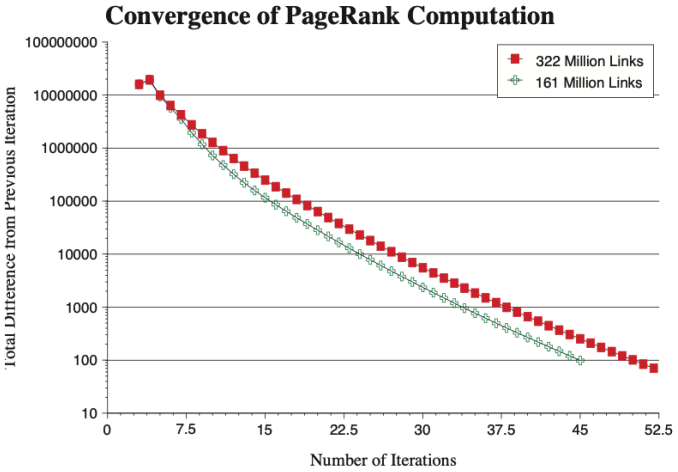
$$R(u;0) = \frac{1}{N}$$

Następnie nasz wzór z czasem t:

$$R(u;t+1) = \frac{1-d}{N} + d \cdot \sum_{b_i \in B_u} \frac{R(b_i;t)}{\deg(b_i)}$$

jest obliczany iteracyjnie dla każdej strony u należącej do grafu, gdzie $R(p;t)$ jest rankingiem strony p w czasie t oraz d jest współczynnikiem tłumienia. Iteracje możemy przerwać gdy wartości PageRank przestaną się zmieniać w stopni znacznym tj. $\big|R(t+1) - R(t)\big| < \epsilon$ dla pewnego ustalonego $\epsilon > 0$.

Założyciele google'a w swoim raporcie [5] opisali, że do obliczenia PageRank'u dla 322 milinów stron wystarczyły 52 iteracje. Dla o połowę mniejszej bazy 42 iteracje, wyciągając wnioski, że współczynnik skalowania jest równy $\log N$.



Rysunek 3.7: Iteracje w obliczeniach PageRank

Macierz PageRank jest łańcuchem Markowa, gdzie z teorii Markowa, jego wartość $R(u)$ jest prawdopodobieństwem przejścia ze strony u z powrotem do strony u po dużej liczbie przejść w błądzeniu losowym (*Random Walk*).

Definicja 3. Łańcuchem Markowa o zbiorze stanów V nazywamy ciąg zmiennych losowych $X_0, X_1, X_2, \ldots$ takich, że

$$\mathbb{P}\big(X_n = v_n \big| X_{n-1} = v_{n-1} \wedge \ldots X_0 = v_0\big) = \mathbb{P}\big(X_n = v_n \big| X_{n-1} = v_{n-1}\big)$$

Oznacza to, że prawdopodobieństwo przejścia jest niezależne od n.

3.2.3. Power Method

Dla szybszego niwelowania błędu w iteracji do obliczania PageRank wykorzystano algorytm Power Method bazujący na wartościach własnych macierzy.

Niech macierz $\mathcal{M} = \left[\mathcal{M}_{i,j}\right]$ będzie macierzą przejścia zdefiniowaną następująco:

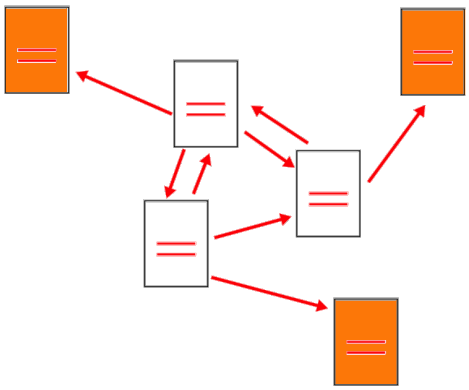
$$\mathcal{M}_{i,j} = \begin{cases} \frac{1}{\deg(p_i)} & \text{jeśli strona } p_i \text{ jest podlinkowana do } p_j \\ 0 & \text{w p.p} \end{cases}$$

oraz PageRank R będzie rozkładem prawdopodobieństwa tj. $\left|R\right| = 1$ i $\mathbf{E}R = 1$ gdzie $\mathbf{E}$ jest macierzą jedynek. Wtedy *Power Method PageRank* jest dany wzorem:

$$R = \left(d\mathcal{M} + \frac{1-d}{N}\mathbf{E}\right) R$$

3.2.4. Dangling Nodes Problem

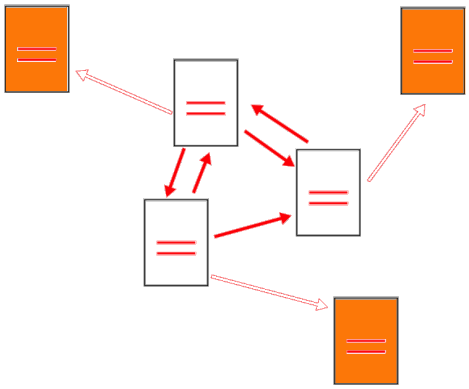
Podczas obliczania PageRank pojawia się *Dangling Nodes Problem* czyli problem stron, które nie posiadają linków wychodzących.



Rysunek 3.8: Przykład grafu z Dangling Nodes

Stawiają kłopotliwe pytanie dla modelu, gdzie ich wartość powinna zostać dalej dystrybuowana. Istnieją dwa rozsądne rozwiązania:

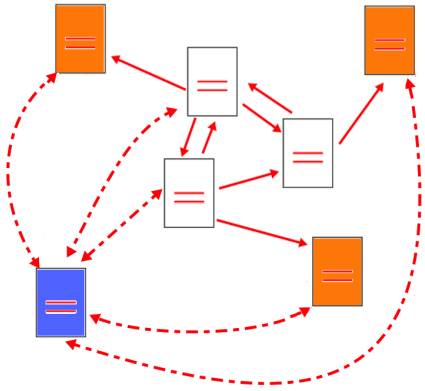
- **Usunąć na czas obliczeń**



Rysunek 3.9: Przykład usuwania Dangling Nodes

Skoro Dangling Nodes nie wpływa bezpośrednio na PageRank żadnej innej strony, możemy je po prostu wykluczyć z naszych obliczeń. Takie strony stanowią dużą część grafu stąd usunięcie ich z obliczeń może okazać się korzystne dla czasu realizacji.

- **Dodać tymczasowy wierzchołek**



Rysunek 3.10: Przykład usuwania Dangling Nodes

Na czas obliczeń możemy dodać tymczasowy wierzchołek, który będzie połączony dwukierunkowo ze wszystkimi wierzchołkami w grafie.

Rozdział 4

Funkcjonalność zaimplementowanego systemu

4.1. Wyszukiwarka

4.2. Wyszukiwanie

4.3. Podstawowe operatory

Rozdział 5

Implementacja oraz użyte technologie

5.1. Java

5.1.1. Apache Lucene

5.2. Spring Boot

5.3. Thymeleaf

5.4. HTML

5.5. Struktura i działanie klas programu

5.5.1. Klasy silnika wyszukiwania

5.5.2. Klasy kontrolera

Rozdział 6

Przypadki użycia

6.1. Uruchomienie aplikacji

6.2. Wyszukiwanie stron

6.3. Ustawianie filtrów

Rozdział 7

Podsumowanie

Bibliografia

[1] Shraddha T. Shela Deepali D. Rane. Review of Lucene Indexing Algorithm on Public Cloud. *International Journal for Research in Applied Science And Engineering Technology*, 2017.

[2] Apache Software Foundation. Class TFIDFSimilarity. `https://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html`, 2000-2012.

[3] Google. How Search Works. `https://www.google.com/search/howsearchworks/crawling-indexing/`, 2010.

[4] Google. Google Search Education Evangelism. `https://sites.google.com/site/gwebsearcheducation/advanced-operators`, 2011.

[5] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[6] Addagada Sridevi. Indexing and Searching Document Collections using Lucene. *University of New Orleans Theses and Dissertations. 1070*, 2007.

[7] 24 Godziny Sp. z o.o. szukacz.pl. `http://szukacz.pl/wiecej_o_szukaczu.html`, 2000-2018.