

# Spis treści

<b>1</b>	<b>Wstęp [TODO]</b>	<b>2</b>
<b>2</b>	<b>Opis aplikacji [TODO]</b>	<b>3</b>
2.1	Folksonomia . . . . .	3
2.2	Architektura . . . . .	3
2.2.1	Zbieranie danych: . . . . .	3
2.2.2	Baza danych . . . . .	5
2.2.3	Interface użytkownika . . . . .	6
<b>3</b>	<b>SocialPageRank</b>	<b>7</b>
3.1	Opis . . . . .	7
3.2	Wyniki algorytmu dla przykładowych danych . . . . .	8
3.3	Implementacja . . . . .	9
3.3.1	Baza danych . . . . .	9
3.3.2	Tworzenie i mnożenie macierzy . . . . .	9
3.4	Wyniki - czesciowe: TODO . . . . .	10
3.4.1	Problemy : TODO . . . . .	10
3.4.2	Przykładowe wyniki wyszukiwarki : TODO . . . . .	10
<b>4</b>	<b>Adapted PageRank</b>	<b>11</b>
4.1	Opis . . . . .	11
4.1.1	Algorytm FolkRank . . . . .	12
4.1.2	Przykładowe wyniki . . . . .	12
4.2	Implementacja . . . . .	13
4.3	Wyniki . . . . .	13
4.4	Porównanie algorytmu SocialPageRank i algorytmu Adapted PageRank : TODO . . . . .	14
4.4.1	Porównanie na małym przykładzie . . . . .	14
4.4.2	Porównanie w działającym systemie . . . . .	14
<b>5</b>	<b>Wyniki [TODO]</b>	<b>15</b>

# Rozdział 1

## Wstęp [TODO]

TODO

## Rozdział 2

# Opis aplikacji [TODO]

### 2.1 Folksonomia

Folksonomia nazywamy krotkę  $F := (U, T, R, Y)$ , gdzie:  $U, T, R$  to zbiory skończone, których elementy składają się odpowiednio z użytkowników, tagów i dokumentów.  $Y$  jest relacją “przypisania tagu” pomiędzy tymi elementami  $Y \subset U \times T \times R$

Użytkownicy i tagi identyfikowani są na podstawie ich unikalnych nazw własnych. Dokumenty mogą być różnymi danymi: stronami www, zdjęciami, plikami np: pliki pdf. Ta praca bazuje na danych pobranych z systemu delicious, które w zdecydowanej większości są stronami www. Dane które nie są stroną www nie są brane pod uwagę w tej pracy.

### 2.2 Architektura

#### 2.2.1 Zbieranie danych:

Dane pobierane są na kilka sposobów. Głównym źródłem nowych danych jest kanał RSS strony delicious. Dodatkowo dla użytkowników i dokumentów które są już zapisane w bazie danych, co powinien przedział czasu, sprawdzane jest czy nie zostały dla nich dodane nowe wpisy na stronie delicious.

#### Nowe dane

Kanał RSS delicious zawiera dane ostatnio dodane przez użytkowników serwisu delicious. Nie są to tylko nowe dane, mogą być to dane które istnieją na stronie, ale zostały dodane ponownie przez innego użytkownika. Każdy wpis zawiera informacje o użytkowniku który ostatnio dodał daną stronę, tagi, adres strony i adres kanału rss tej strony.

1. Crawler Delicious:

- sprawdza najnowsze dane dodawane przez wszystkich użytkowników na głównej stronie delicious.
- sprawdza popularne strony dodawane przez użytkowników. To czy dana strona znajdowała się wśród popularnych jest również zapisywanie w bazie danych.
- update: sprawdzanie danych ze strony użytkowników którzy już są dodani jak również sprawdzanie czy strony które już były dodane nie otrzymały nowych tagów i czy nie było innych użytkowników którzy dana stronę dodali również (pozwala to na szybkie zwiększenie danych, ale również może spowodować potencjalne problemy: może to spowodować że dane które będą w bazie będą do siebie podobne - ci sami użytkownicy dodają podobne strony, w podobnej tematyce, z drugiej strony, osoby którzy dodali daną stronę, mogą mieć podobne preferencje)

Najnowsze dane pochodzą z przeglądania głównego RSS'a strony. Po sparowaniu danych, wyciągane są nowe strony. Każdy z tych nowych URL'ów posiada swoją stronę na delicious, na której przechowywane są dane o użytkownikach którzy dodali i tagach użytych. Przeglądane zostaje

#### 2. Crawlery innych serwisów społecznościowych:

w swoim systemie korzystam z serwisów twitter, facebook, digg. Dostarczają one API które pozwalają sprawdzić ile użytkowników udostępniło daną stronę. Działają one niezależnie od siebie. Sprawdzane są jednocześnie nowo dodane do systemu strony jak również przeprowadzany jest update pozostałych

#### 3. Lucene:

Zadaniem tego crawlera jest sparowanie nowo dodanych stron do systemu, i ściąganie ich treści na dysk. Po ściągnięciu danej strony, usuwane są z niej wszystkie znaczniki HTML, a następnie zapisywane są na dysku przy pomocy frameworku lucene.

Funkcja lucene:

$$score(q, d) = coord(q, d) * queryNorm(q) * \sum_{t \text{ in } q} \left( tf(t \text{ in } d) * idf(t)^2 * getBoost(t) * norm(t, d) \right)$$

#### 4. cache:

- statystyki – co pewien określony czas na bazie danych robiony jest update statystyk. Wyliczanie statystyki powodują jednak duże obciążenie bazy danych. Największym problemem nie jest obciążenie ale obecnie wyliczane statystyki:
  - dla użytkownika:
    - \* ilość dodanych dokumentów



### Opis tabel:

- USER: tabela zawiera dane użytkowników, ich *id* i unikalną nazwę pobraną z serwisu delicious.
- DOCUMENT: tabela zawiera dane o dokumentach. *URL* który reprezentuje dokument jest unikalny. Ponieważ adresy stron po pominięciu ostatniego slashu/backslasha prowadzą do tych samych witryn, przy sprawdzaniu unikalności linku brane pod uwagę są wszystkie kombinacje adresów.
- TAG: tabela zawierająca adnotacje stron.
- USERTAGDOC i USERTAGDOC\_TAG: są to tabele które służą do zapisania w bazie danych relacji nadania  $k$  tagów:  $t_n, t_{n+1}, \dots, t_{n+k}$  przez użytkownika  $user_j$  dokumentowi  $doc_m$ . Dodatkowo wartość  $k$ : ilość przypisanych tagów, zapisana jest w polu *count*.
- TAG\_USER i TAG\_DOC: tabele zawierające redundantne dane. TAG\_USER zawiera informacje o ilości dokumentów opisanym tagiem  $tag_k$  przez użytkownika  $user_n$ . Druga tabela zawiera informacje o liczbie przypisań tagu  $tag_k$  do dokumentu  $doc_m$ . Tabele te są wykorzystywane dla szybszego zbierania danych dla algorytmów Adapter PageRank i SocialPageRank.

### 2.2.3 Interface użytkownika

interface użytkownika napisany jest przy pomocy technologii google-web-toolkit. główny panel zawiera u góry zakładki, które pozwalają na przemieszczenie się między wyszukiwarką, statystykami, tagami, innymi informacjami.

użytkownik po wczytaniu zapytania i naciśnięciu ENTER otrzymuje wyniki zapytania.

### Tagi:

Tagi nie są dodawane do bazy danych w postaci dokładnie uzyskanej ze strony delicious. Wiele z nich wymaga paru przekształceń. Podstawowym jest usunięcie białych znaków z początku i końca słowa. Dodatkowo, większość tagów, zaczyna lub kończy się na znakach specjalnych, lub znajduje się w cudzysłowie czy też kończy się znakiem przecinka. Dla przykładu:

- @java
- @@java
- #java
- java@
- "tag" / "tag / tag"
- tekst, / ,tekst

## Rozdział 3

# SocialPageRank

### 3.1 Opis

SocialPageRank jest statycznym rankingiem stron z perspektywy użytkownika sieci. Algorytm bazuje na obserwacji relacji między popularnymi stronami, tagami i udzielającymi się użytkownikami. Popularne strony są dodawane przez udzielających się użytkowników, które są opisywane popularnymi tagami. Udzielający się użytkownicy używają popularnych tagów dla popularnych stron. Popularne tagi używane są do annotacji popularnych stron przez ważnych użytkowników.

Bazując na powyższych założeniach algorytm propaguje i wzmacnia zależności między popularnymi tagami, użytkownikami i dokumentami.

#### Dane wejściowe:

$N_T$ : ilość tagów

$N_U$ : ilość użytkowników

$N_D$ : ilość dokumentów

$M_{DU}$ : macierz  $N_D \times N_U$  asocjacyjna między dokumentami a użytkownikami

$M_{UT}$ : macierz  $N_U \times N_T$  asocjacyjna między użytkownikami a tagami

$M_{TD}$ : macierz  $N_T \times N_D$  asocjacyjna między tagami a dokumentami

$P_0$ : wektor, od długości  $N_D$ ,

#### Inicjalizacja

W komórce macierzy  $M_{DU}(d_n, u_k)$  znajduje się wartość będąca ilością annotacji przypisanych do dokumentu  $d_n$  przez użytkownika  $u_k$ . Podobnie dla pozostałych macierzy, elementy  $M_{UT}(u_k, t_n)$  to ilość dokumentów opisanych tagiem  $t_n$  przez użytkownika  $u_k$ , elementy  $M_{TD}(t_n, d_k)$ : ile użytkowników dodawało dokument  $d_k$  i oznaczyło go annotacją  $t_n$ .

Wektor  $P_0$  zainicjalizowany został losowymi wartościami z przedziału  $[0, 1]$ . Jest on pierwszym przybliżeniem rank dokumentów.

```

repeat
   $U_i = M_{DU}^T * P_i$ 
   $T_i = M_{UT}^T * U_i$ 
   $P'_i = M_{TD}^T * T_i$ 
   $T'_i = M_{TD} * P'_i$ 
   $U'_i = M_{UT} * T'_i$ 
   $P_{(i+1)} = M_{DU} * U'_i$ 
until wartości wektora  $P_n$  nie zbiegną

```

### Złożoność

Złożoność czasowa każdej iteracji wynosi  $O(N_u * N_d + N_t * N_d + N_t * N_u)$ .

## 3.2 Wyniki algorytmu dla przykładowych danych

W poniższej tabelce znajdują się dane, dla których zostało sprawdzone działanie algorytmu Social PageRank. Dane są nie duże i składają się z trzech różnych dokumentów, dwóch użytkowników i trzech tagów.

Strony www	użytkownicy	
	użytkownik 1	użytkownik 2
http://www.ted.com/	inspiration	
http://www.colourlovers.com/	design	inspiration
http://www.behance.net/	portfolio, design	portfolio, inspiration

Dla takich danych macierz  $M_{d,u}$  mówiąca o zależności dokumentów z użytkownikami ma postać:

$$M_{d,u} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix}$$

Macierz użytkowników i tagów,  $M_{u,t}$ :

$$M_{u,t} = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 0 & 1 \end{pmatrix}$$

Macierz tagów i dokumentów,  $M_{t,d}$ :

$$M_{t,d} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{pmatrix}$$

### wyniki:

Dla powyższych danych wyniki algorytmu zbiegają po czterech iteracjach z dokładnością  $|P_3 - P_4| < 10^{-10}$ . Wyniki zostały przedstawione w poniższej tabelce:



	Social PageRank
www.ted.com	0.2381373691295440
www.colourlovers.com	0.4343479235414989
www.behance.net	0.8686958470829979

Można zauważyć, że największy ranking ma strona behance.net która została dodana przez dwóch użytkowników i oznaczonych najpopularniejszymi tagami - 2 razy tagiem portfolio, użytym tylko dla tej strony, raz tagiem design, który użyty był 2 razy w powyższych danych i również raz tagiem inspiration, który jest najpopularniejszym tagiem, użytym w przykładzie aż 3 razy.

### 3.3 Implementacja

Algorytm pozwala na wcześniejsze wyliczenie rankingu dlatego został zaimplementowany jako osobny proces działający w określonych okresach czasu. Dodatkowo algorytm wymaga danych, które muszą być wyliczone i zapisane w bazie danych przed rozpoczęciem jego działania.

#### 3.3.1 Baza danych

Poniżej znajduje się obrazek przedstawiający bazę danych ze zmianami wymaganymi dla sprawnego działania algorytmu. Dodatkowe tabele dodane zostały dla przyspieszenia generowania danych wejściowych. Dane w nich zawarte wyliczane są z danych wyliczane są z danych już istniejących w bazie danych.

TODO: OBRAZEK - BAZA DANYCH Z DODATKOWYMI TABELAMI I ZAZNACZONYMI UŻYWANYMI POLAMI W BAZIE DANYCH

Tabela DOKUMENT zawiera dodatkowo pole soc\_page\_rank służące do przechowywania wyników algorytmu. Tabele TAG\_USR i TAG\_DOC jak również pole w tabeli USERTAGDOC.how\_much zawierają redundantne dane wykorzystywane do generowania macierzy. Pole USERTAGDOC.how\_much zawiera informacje o ilości tagów użytych przez użytkownika do opisanie dokumentu. TAG\_USR zawiera relacje między użytkownikami i tagami, ilość annotowanych dokumentów przez tą parę znajduje się w polu how\_much. Analogicznie TAG\_DOC jest relacją między anotacjami a dokumentami z ilością ich wykorzystania.

#### 3.3.2 Tworzenie i mnożenie macierzy

Algorytm wymaga w każdej iteracji wykorzystania sześć macierzy. Z powodu wielkości danych nie jesteśmy w stanie przechowywać ich wszystkich w pamięci. Dodatkowo biblioteka wykorzystana do mnożenia macierzy nakłada ograniczenia na ilość kolumn i wierszy w macierzach. Kolejnym problemem jest czas potrzebny na pobranie danych z bazy danych.

Z powodu tych ograniczeń macierze pobierane są do partiami do pamięci z bazy danych, przetwarzane i zapisywane są w struktury ułatwiające szybki do

nich dostęp. Następnie, zapisywane są w plikach zawierające największe porcje danych mieszczące się jednocześnie w pamięci.

W każdej iteracji algorytm pobiera dane z dysku. Z tych danych tworzona jest macierz o maksymalnej możliwej ilości wierszy na które pozwala wykorzystana biblioteka. Każda z tych części jest osobno mnożona przez wektor. Wyniki następnie składane są w wektorze wynikowym, który przekazywany jest do kolejnego mnożenia macierzy

#### WYKRES POKAZUJĄCY PROCES TWORZENIA I MNOŻENIA CZĘŚCIOWEGO MACIERZY

Mimo, że wymagania pamięciowe sprawiają, że trzeba wykonać większą liczbę działań w czasie mnożenia, macierze generowane są dość rzadkie. Większość pól zawiera wartość 0, co przyspiesza mnożenie macierzy i wektorów.

TODO: ilość wykorzystanych plików przy prawdziwych danych (1 mln)

TODO: ilość mnożeń

TODO: wykorzystana biblioteka: cern.colt

### 3.4 Wyniki - czesciowe: TODO

// wyniki dla danych 66 000 dokumentów - TODO - wyniki dla duzych danych

Strona <http://www.pythonchallenge.com/> jest jedną ze stron z największym wynikiem socialpagerank ( 0.00301). Dodane jest przez 785 różnych użytkowników. Zostało użyte do tego 209 unikalnych tagów. Najpopularniejsze tagi, w kolejności od najczęściej użytego to python, programming, challage i puzzle.

jedną ze stron o najniższym rankingu jest np: <http://djangosnippets.org/snippets/1314/>. Strona ta zawiera specyficzne rozszerzenie dla frameworku django. Można się spodziewać że nie będzie to popularna witryna. Została ona dodana przez jednego użytkownika i opisana siedmioma anotcjami.

Strony które uzyskały ranking 0 to strony które nie zostały opisane żadnymi tagami przez użytkowników.

#### 3.4.1 Problemy : TODO

Potencjalne problemy zauważone na mniejszej ilości danych: algorytm jest podatny na cykle które mogą zostać stworzone przez dużą ilość wygenerowanych użytkowników.

#### 3.4.2 Przykładowe wyniki wyszukiwarki : TODO

TODO: OPISAC POZNIEJ, PO ZAIMPLEMENTOWANIU WYSZUKIWANIA UZYWAJACEGO SOCIALPAGERANK

## Rozdział 4

# Adapted PageRank

### 4.1 Opis

Algorytm Adapted Page Rank jest zainspirowany algorytmem PageRank. Ideą za algorytm PageRank jest pomysł, że strona jest ważna jeśli dużo innych stron ma odnośniki wskazujące na tą stronę i te strony są również ważne.

Ponieważ algorytm page rank nie może być bezpośrednio zastosowany do zebranych danych autorzy algorytmu adapted page rank zmienili strukturę danych na nieskierowany graf trydzienny  $G_f = (V, E)$ .

#### Proces tworzenia grafu $G_f$ :

1) zbiór wierzchołków  $V$  powstaje z sumy rozłącznej zbioru użytkowników, tagów, i dokumentów:  $V = U \cup T \cup R$

2) wszystkie wystąpienia łącznie tagów, użytkowników, dokumentów stają się krawędziami grafu  $G_f$ :  $E = \{\{u, t\}, \{t, r\}, \{r, u\} | (u, t, r) \in Y\}$ . Wagi tych krawędzi są przydzielane w następujący sposób: każda krawędź  $\{u, t\}$  ma wagę  $|\{r \in R : (u, t, r) \in Y\}|$ , czyli jest to ilość dokumentów, którym użytkownik  $u$  nadał anotację  $t$ . Analogicznie dla krawędzi  $\{t, r\}$ :  $waga = |\{u \in U : (u, t, r) \in Y\}|$  i krawędzi  $\{r, u\}$ , gdzie  $waga = |\{t \in T : (u, t, r) \in Y\}|$ .

#### dane wejściowe:

$A$  – prawo stochastyczna macierz sąsiedztwa grafu  $G_f$

$p$  – wektor preferencji

$w$  – losowo zainicjalizowany wektor

$\alpha, \beta, \gamma$  – stałe, gdzie:  $\alpha, \beta, \gamma \in [0, 1]$  i  $\alpha + \beta + \gamma = 1$

do:

$w = \alpha * w + \beta * A * w + \gamma * p$  while: do czasu kiedy wartości wektora  $w$  zbiegną się.

wynikiem algorytmu jest wektor  $w$ .

Dane uzyskane w systemie zostały uzyskane dla parametrów:  $\alpha = 0.35, \beta = 0.65, \gamma = 0$  i pochodzą z pracy (cytat).

W algorytmie adapted page rank wektor preferencji  $p = 1$ .

#### 4.1.1 Algorytm FolkRank

Algorytm FolkRank jest wersją algorytmu adapted page rank, w którego daje różne wyniki w zależności od tematu. Temat tej jest ustalany w wektorze preferencji  $p$ . Wektor preferencji może być ustalony na wybrany zbiór tagów, użytkowników, dokumentów, albo na pojedynczy element. Wybrany temat będzie propagowany na reszcie dokumentów, tagów i użytkowników. Można dzięki temu, ustalając wagę na zapytanie albo konto użytkownika systemu uzyskać wyniki bardziej zbliżone do zainteresowań danego użytkownika.

Algorytm FolkRank może pomóc w analizie zbioru danych, albo w systemach nie działających w czasie rzeczywistym, ale nie jest użyteczny w zaprezentowanym systemie nie jest możliwy do wykorzystania z powodu długiego czasu obliczania wag dokumentów.

#### 4.1.2 Przykładowe wyniki

Działanie algorytmu dla danych złożonych z 3 różnych dokumentów, 2 użytkowników i 3 tagów.

Strony www	użytkownicy	
	użytkownik 1	użytkownik 2
http://www.ted.com/	inspiration	
http://www.colourlovers.com/	design	inspiration
http://www.behance.net/	portfolio, design	portfolio, inspiration

macierz asocjacyjna powstała z powyższych danych ma wymiary  $8 \times 8$  i wygląda:

$$G_f = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 2 & 2 & 1 & 1 & 2 \\ 1 & 1 & 2 & 0 & 0 & 1 & 2 & 1 \\ 0 & 1 & 2 & 0 & 0 & 2 & 0 & 1 \\ 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

	Social PageRank
doc: http://www.ted.com/	0.280676409730572
doc: http://www.colourlovers.com/	0.369220441236174
doc: http://www.behance.net/	0.384551423972747
usr: użytkownik A	0.402473669662513
usr: użytkownik B	0.354578057974890
tag: inspiration	0.383291185401107
tag: design	0.321455285546253
tag: portfolio	0.314739469615726

Zbierność wektora została uzyskana po 22 iteracjach.

Patrząc na powyższe wyniki najwyższy ranking wśród dokumentów ma strona beance.net: została ona dodana przez 2 użytkowników i przypisane jej zostały 4 tagi. Niewiele niższy ranking ma witryna colourlovers dodana przez 2 użytkowników i opisana 2 różnymi anotacjami. Można po tym wywnioskować że nadanie większej ilości tagów nie ma dużego wpływu na rank strony. Za to zmniejszenie liczby użytkowników którzy tą stronę dodali, ma duże: przykład strona ted.com i colourlovers.com gdzie widoczny jest dość duży skok wartości wyniku.

## 4.2 Implementacja

Z powodu długiego czasu działania i dużej ilości wymaganych danych nie mogą być one pobierane bezpośrednio z bazy danych. Przed rozpoczęciem działania algorytmu są one pobierane, zapisywane w struktury pozwalające na lepsze i szybsze ich przeglądanie i serializowane do pliku. Dodatkowo żeby przyspieszyć tworzenie danych korzystamy z dodatkowych tabel zawierających już wyliczone dane np: o ilości dokumentów dodanych i opisanych tym samym tagiem przez użytkowników.

W czasie każdej iteracji algorytmu są one pobierane z pliku, zamieniane na macierze i poddawane dalszym operacjom. Po zakończeniu działania algorytmu wyniki zapisywane są w bazie danych. Z powodu tego, że dane wymagane w algorytmie adapted pagerank są zbliżone do danych wymaganych w algorytmie socialpagerank wykorzystywane są te same zserializowane dane z bazy danych jak również. Dokładniej macierz wygląda następująco:

$$G_f = \begin{pmatrix} 0 & M_{UD} & M_{TD}^T \\ M_{DU}^T & 0 & M_{UT} \\ M_{TD} & M_{UT}^T & 0 \end{pmatrix}$$

Dodatkowo podobnie jak w algorytmie socialPageRank z powodu wielkości macierzy i ograniczeń pamięciowych są one mnożone częściowo przez wektor wag.

## 4.3 Wyniki

// TODO: po zebraniu wystarczającej ilości danych

// opis kilku przykładowych dokumentów

## 4.4 Porównanie algorytmu SocialPageRank i algorytmu Adapted PageRank : TODO

### 4.4.1 Porównanie na małym przykładzie

#### Szybkość działania:

Zbierność wektora uzyskano szybciej - bo już po pięciu iteracjach przy algorytmie SocialPageRank. W algorytmie Adapted Page Rank wymagało to aż 22 iteracji. Zbierność można przyspieszać przez zmianę parametru alpha w algorytmie page rank.

#### Wyniki:

Kolejność wag dla dokumentów jest taka sama w przypadku jednego i drugiego algorytmu, ale ich wartości są zdecydowanie inne. W przypadku dokumentu o największej randze: behence.net i kolejnego colourlovers.com różnica dla algorytmu adapted page rank jest niewielka proporcjonalnie do wagi, a przy social page rank waga behence.net jest prawie dwa razy większa od poprzednika. Widoczna podobieństwo jest dopiero dla dokumentów colourlovers.com i ted.com gdzie różnica dla jednego i drugiego algorytmu jest znaczna. Jest to prawdopodobnie spowodowane, że na wynik jednego i drugiego algorytmu wpływa ilość użytkowników którzy dodali dany dokument, za algorytm SocialPageRank bierze bardziej pod uwagę ilość anotacji przypisanych przez użytkowników danym dokumentom.

#### Dodatkowe dane:

Adapted Page Rank daje nam dodatkową wiedzę w postaci rank dla tagów i użytkowników. O ile nie jest to przydatne przy wyszukiwaniu, ale daje dodatkowe informacje o posiadanych danych.

### 4.4.2 Porównanie w działającym systemie

// TODO: zrobić po zebraniu dużej ilości danych  
// wykresy ??  
// porównanie i opis dla kilku dokumentów

## Rozdział 5

# Wyniki [TODO]

TODO