Spis treści

Rozdział 1

Wstęp [TODO]

TODO

Rozdział 2

Opis aplikacji [TODO]

2.1 Folksonomia

TA DEFINICJA MA BYC W INNY MIEJSCU:

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 $Y \subseteq U \times T \times R$

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 witryny delicous, które w zdecydowanej większości są stronami www. Dane które nie są stroną www nie są brane pod uwagę w tej pracy.

2.2 delicious.com : TODO

TODO: opis strony delicious

2.3 Architektura

2.3.1 Wątki

Główne zadania w serwisie wykonywane są przez odpowiednie wątki. Część z nich działa cały czas, część z nich reaguje na odpowiednie zmiany w bazie danych, a inne uruchamiane są na żądanie. Poniżej jest lista głównych zadań wykonywanych przez te procesy. Dokładne opisy znajdują się w kolejnych rozdziałach:

- RecentCrawler: przegląda strone delicious.com i zapisuje nowo dodane strony, użytkowników i tagi

- URLCrawler: sprawdza czy nie nastąpiła jakaś zmiana w dokumentach zapisanych już w naszej bazie danych np: czy nie zostały one dodane przez większą liczbę użytkowników
- UserCrawler: zadaniem tego crawlera jest sprawdzenie czy użytkownicy, o których informacje posiadamy nie dodali innych stron do swoich kolekcji dokumentów.
- SocialNetowrkCrawler: ten wątek nie pobiera danych z serwisu delicious, ale sprawdza czy URL'e zapisane w bazie danych zostały dodane przez użytkowników innych serwisów, takich jak: digg, twitter, facebook.
- Procesy odpowiedzialne za algorytmy: każdy z algorytmów (social pagerank, adapted pagerank) posiada własny wątek, który przygotowuje dla niego dane, a następnie uruchamia działanie samego algorytmu.
- cache: wątki te odpowiadaja są za wyliczanie danych, dzieki czemu później jest do nich szybszy dostpęp. Naprzykład w tabeli DOCUMENT dodawane są również dodatkowe informacje, które są wypisywane w momencie kiedy wyszukiwarka zwróci wynik dla użytkownika.Wyliczanie tych informacji w trakcie wyszukiwania byłoby zbyt czasochłonne. Dane zapisywane tam to np: kilka najczęściej używanych tagów dla tego dokumentu wraz z ich ilością i ilość uzytkowników którzy dodali ten dokument.

2.3.2 Dane

Dane pobierane są na kilka sposobów. Głównym źródłem nowych informacji jest kanał RSS strony delicious. Serwis delicious daje użytkownikom dostęp do kilku kanałów RSS. Kanał 'recent' zawierający najnowsze dokumenty, dodawane właśnie przez użytkowników. Dodatkowo swój kanał RSS zawierają też użytkownicy i adnotacje - znajdują się w nich ostatnio dodane przez danego użytkownika dokumenty, a w kanale RSS adnotacji: ostatnio opisane danym tagiem dane. Każdy dokument zawiera także własny kanał RSS, znajdują się w nim informacje o użytkownikach i tagach którym został opisany dany dokument.

Nowe dane

Kanał delicious zawierający najnowsze dokumenty wykorzystywany jest do pobierania informacji o ostatnio dodanych dokumentach. Kilka wątków w aplikacji sprawdza na bieżąco stronę. Dla każdego dokumentu znajdującego się w pobranych danych, sprawdzany jest jej kanał RSS i stamtąd pobierane są dane o użytkownikach i tagach którymi dany dokument został opisany. Następnie wszystkie te informacje zapisywane są w bazie danych.

Odświerzanie danych

W czasie działania wątków pobierających nowe dane działają również wątki sprawdzające czy nie zostały dodane nowe wpisy przez zapisanych już użytkowników, albo czy informacje posiadane o dokumencie nie zmieniły się: np dodany został również przez innego użytkownika i opisany innymi tagami. Wykorzystywane do tego są kanały RSS użytkowników i dokumentów.

Czyszczenie danych

Dane pobierane z serwisu delicous nie zawsze są w postaci wymaganej przez aplikacje. Spowodowane to jest błędami użytkowników, albo specificznym stylem zapisywania tagów przez nich.

Niektóre tagi mają różne znaczenie w zależności od kontekstu, na przykład tag 'design' ma inne znaczenie w kontekście strony o programowaniu, a inne w kontekście strony o sztuce. Część użytkowników żeby poradzić sobie z tym problemem dodają do tagów informacje mówiące o ich domenie. Często domena ma wygląd 'programming@design' czy 'art#design'. Żeby rozwiązać ten problem adnotację przed dodaniem do bazy danych są dzielone ze względu na najpopularniejsze znaki specjalne. Dodane kontekstu do tagu mogłoby być przydatne w aplikacji, ale z powodu tego że każdy użytkownik ma swój specyficzny sposób opisywania dokumentów np: design@art i art#design, trudno jest je zunifikować. Dodatkowym problemem jest to, że nie jest to sposób opisu używany przez wszystkich użytkowników.

Adnotacje przypisywane przez użytkowników często kończą się lub zaczynają od znaków specjalnych. Jest to spowodowane np: błędami (dodatkowe przecinki) albo specyficznym stylem opisywania danych przez użytkownika. Wszystkie znaki specjalne z końca i początku dokumentu są usuwane przed dodaniem do bazy danych.

Przykłady danych przed i po ich oczyszczeniu:

- '@java' : 'java'
- '@@java' : 'java'
- '#java6@' : 'java6'
- 'design!$%@art' : 'design', 'art'
- 'art!#,': 'art'

Dane z innych serwisów społecznościowych

W systemie wykorzystywane są również dane z innych serwisów społecznościowych: twitter, facebook i digg. Dostarczają one API, które pozwala sprawdzić ile użytkowników udostępniło dana stroną w tych serwisach. Dla

każdego z tych serwisów tworzony jest osobny wątek, który sprawdza czy w bazie danych nie ma nowo dodanych dokumentów. Dla tych dokumentów sprawdzany jest odpowiedni serwis, wynik zapisywany jest w bazie danych w tabeli DOCUMENT. Co pewien okres czasu przeprowadzane jest sprawdzanie wszystkich dokumentów i informacje są nadpisywane.

Przykładowe dane z bazy danych TODO

TODO

2.3.3 Dane testowe TODO

TODO - wiecej/poprawic

Żeby zapewnić unikalność danych testowych dane te pochodzą ze strony arXiv.org. Strona ta zawiera publikację z dziedzin fizyki, metamatyki, biologii, ... Z powodu specyfiki tych dokumentów i raczej małej ich popularności w serwiwach społecznościowych można stworzyć z nich unikalne dane testowe.

Jako tekst który jest przekazywany do frameworku lucene używany jest tekst publikacji wyciągniety z pliku pdf.

Tagi: są sumą słów kluczowych podanych w publikacji, tytułu i nazw działów do których dany dokument został przypisany.

TODO - wiecej/poprawic

Opis: TODO

TODO

Przykładowe dane testowe: TODO

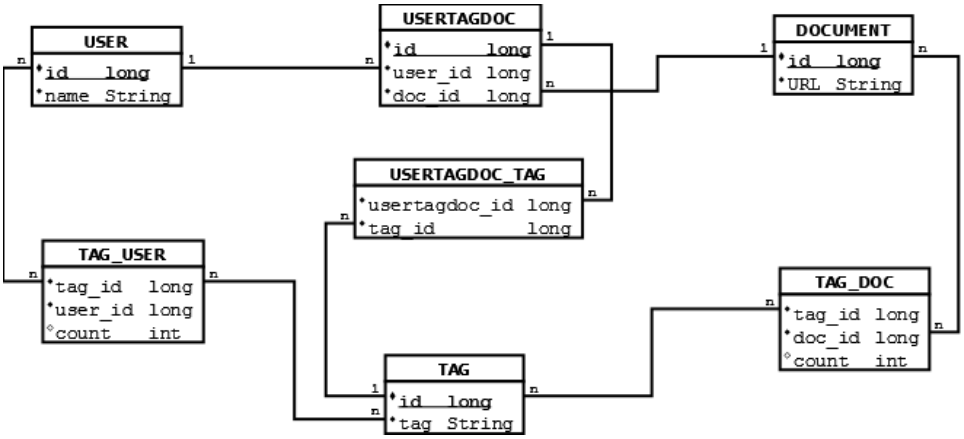
TODO

2.4 Baza danych

Jako serwer bazy używany jest MySQL 5.1. Komunikacja między aplikacją a bazą danych odbywa się za pomocą frameworku Hibernate. Framework ten zapewnia translację danych z relacyjnej bazy danych na obiekty używane w aplikacji.

Opis tabel:

- USER: tabela zawiera dane użytkowników, ich *id* i unikalną nazwę pobraną z serwisu delicous.
- DOCUMENT: tabela zawiera dane o dokumentach. URL który reprezentuje dokument jest unikalny. Ponieważ adresy stron po pominięciu



Rysunek 2.1: Baza danych

ostatniego slasha/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: $t_n, t_{n+1}, \ldots, t_{n+k}$ przez użytkownika $user_j$ dokumentowi doc_m. Dodatkowo wartość k: ilość przypisanych tagów, zapisana jest w polu *count* tabeli USERTAGDOC.
- TAG_USER i TAG_DOC: tabele zawierające redundantne dane przyśpieszające wykonywanie innych operacji. TAG_USER zwiera 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 głównie dla szybszego zbierania danych dla algorytmów Adapter PageRank i SocialPageRank.

2.5 Interface użytkownika TODO

TODO: więcej

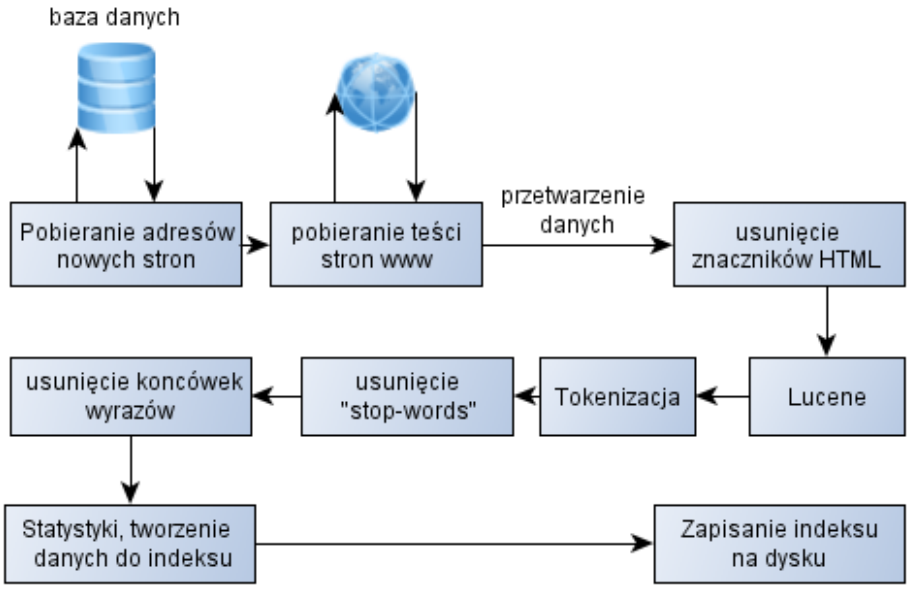
Interface uzytwkonika napisany jest przy pomocy technologi google-web-toolkit. Użytkownik po wczytaniu zapytaniu i nacisnieciu ENTER otrzymuje wyniki zapytania.

2.6 Lucene

Lucene jest biblioteką napisaną w Javie. Biblioteka ta jest w stanie indeksować dużą ilość dokumentów z różnych źródeł i przeprowadzać wyszukiwania w tych tekstach. W opisywanej aplikacji, framework Lucene przechowuje źródła stron, o których informacje zostały pobrane z serwisu delicous i zapisane w bazie danych.

2.6.1 Pobieranie stron

W pewnych odstępach czasu, wątek odpowiedzialny za indeksowanie stron sprawdza czy w bazie danych tabela DOCUMENT nie ma informacji o nowych wpisach. Z bazy danych pobrane są informacje o adresach tych stron. Następnie dla każdego adresu URL zostaje pobrana treść strony na którą wskazuje. Strona WWW następnie zostaje oczyszczona ze znaczników HTML, i przekazane do frameworku lucene do zaindeksowania. Jeśli wszystkie czynności zakończą się powodzeniem, w bazie danych zostaje odnotowana informacja o posiadaniu na dysku danego dokumentu. Rysunek 2.2 przedstawiony jest cały proces pobierania i przetwarzania danej strony.



Rysunek 2.2: Lucene: pobieranie danych i indeksowanie

Do Lucene zapisywane są następujące informacje: identyfikator *id* dokumentu z bazy danych, oraz przetworzony tekst strony WWW. Przechowywanie identyfikatora dokumentu w danych Lucene pozwala późniejsze powiązanie wyników wyszukiwania z odpowiednim rekordem w bazie danych.

W czasie indeksowania biblioteka Lucene wykonuje wiele czynności które pozwalają jej później szybko wyszukiwać informację. Główne z nich to:

- Tekst zostanje przetworzony na ciąg termów,
- usunięcie końcówek wyrazów,
- usunięcie 'stop-words' z tekstu, czyli słów nie mających dużego znaczenia przy wynikach wyszukiwania,
- obliczenie statystyk, np: wystąpienia słów w dokumencie, odległości od siebie

2.6.2 Wyszukiwanie

Czas wyszukiwania zapytania w dokumentach przechowywanych Lucene jest szybkie. Przy małej, poniżej 1GB danych, wyszukiwanie następuje praktycznie w czasie rzeczywistym. Zapytanie jest przekazywane do frameworku, w którym jest one przekształcane na termy. Dla zapytania q i dla każdego dokumentu d wyliczana jest wartość funkcji $score(q, d)$. Wynikiem są dokumenty posortowane wg. wyniku tej funkcji.

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

gdzie:

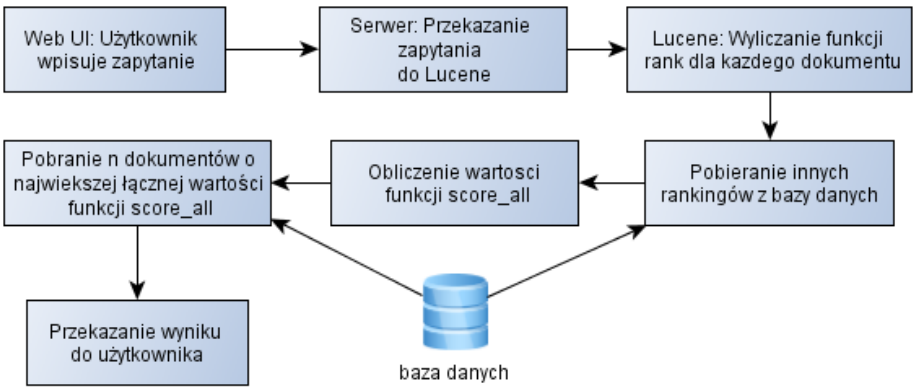
- coord(q,d): funkcja zwraca wartości zależne od miejsca wystąpowania i odległości od siebie szukanych termów w dokumencie.
- queryNorm(q): funkcja normalizująca wyniki zapytania
- tf(t in d): funkcja wyliczająca częstość występowania danego termu w dokumencie
- idf(t) : funkcja wyliczająca częstość występowania termu we wszystkich dokumentach.
- getBoost(t) - Lucene pozwala na zwiększenie wagi niektórych termów. Nieużywane w tej aplikacji.

Lucene ocenia dokumenty głównie na podstawie funkcji TF-IDF. Każdy dokument reprezentowany jest przez wektor, składający się z wag słów występujących w tym dokumencie. TFIDF informuje o częstości wystąpienia termów uwzględniając jednocześnie odpowiednie wyważenie znaczenia lokalnego termu i jego znaczenia w kontekście pełnej kolekcji dokumentów.

2.7 Wyszukiwanie TODO

$score\_all(doc, q) = \alpha * lucene\_score(doc, q) + \beta * social\_page\_rank\_score(doc) + \gamma * adapt\_page\_rank(doc) + \eta * social\_rank(doc)$
gdzie:

- $\alpha, \beta, \gamma, \eta$ wspolczynniki wag dla poszczególnych algorytmów
- *lucene_score* - rank dokumentu *doc* dla zapytania q. Zwracane przez framework lucene.
- *social_page_rank* wynik algorytmu social page rank dla dokumentu *doc*
- *adapt_page_rank* wynik algorytmu adapted page rank
- *social_rank* wynik algorytmu social rank, działającego na danych pobranych z serwisu digg/twitter/facebook



Rysunek 2.3: Proces wyszukiwania dokumentów i wyliczania ich rank

Rozdział 3

SocialPageRank

3.1 Opis

SocialPageRank jest algorytmem wyliczającym statyczny ranking stron z perspektywy użytkownika sieci. Algorytm bazuje na obserwacji relacji miedzy popularnymi stronami, tagami i aktywnymi użytkownikami. Popularne strony są dodawane przez aktywnych się użytkowników, które są opisywane popularnymi tagami. Aktywni 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 wejsciowe:

N_T: ilośc tagów
N_U: ilośc użytkowników
N_D: ilośc dokumentów
M_{DU}: macierz $N_D \times N_D$ 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, o długości N_D,

Inicjalizacja

W komórce macierzy$M_{DU}(d_n, u_k)$ znajduje się wartość będąca ilością adnotacji 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 zainicializowany 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

Złożność

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 sie dane, dla których zostało sprawdzone działanie algorytmu Social PageRank. Dane są nie duże i składają sie 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:

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

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

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

Maciez 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 dokładnościa $|P_3 - P_4| < 10^{-10}$. Wyniki zostały przedstawione w poniższej tabelce:

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

Można zauwazyć, że największy ranking ma strona behance.net ktora 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 wczesniejsze wyliczenie rankingu dlatego został zaimplementowany jako osobny proces. 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 przestawiający bazę danych ze zmianami wymaganymi dla sprawnego działania algorytmu. Dodatkowe tabele dodane zostały dla przyśpieszenia 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 UZYWANYMI POLAMI W BAZIE DANYCH

Tabela DOKUMENT zawiera dodatkowo pole soc_page_rank służące do przechowywania wyników algorytmu. Tabele TAG_USR i TAG_DOC jak równiez pole w tabeli USERTAGDOC.how_much zawieraja redundantne dane wykorzystywane do generowania macierzy. Pole USERTAGDOC.how_much zawiera informacje o ilości tagów użytych przez użytkownika do opisania dokumentu. TAG_USR zaiera relacje miedzy użytkownikami i tagami, ilość annotowanych dokumentów przez tą parę znajduje się w polu how_much. Analogicznie TAG_DOC jest relacja między annotacjami 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 częsci jest osobno mnożona przez wektor. Wyniki następnie składane są w wektorze wynikowym, który przekazywany jest do kolejnego mnożenia macierzy

WYKRES POKAZUJACY PROCES TWORZENIA I MNOZENIA CZĘŚCIOWEGO MACIERZY

Mimo, że wymagania pamięciowe sprawiają, że trzeba wykonać wiekszą liczbe działań w czasie mnożenia, macierze generowane są dość rzadkie. Wiekszość pól zawiera wartość 0, co przyśpiesza mnożenie macierzy i wektorów.

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

TODO: ilość mnożen

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 najwiekszym wynikiem socialpagerank (0.00301). Dodane jest przez 785 różnych użytkowników. Zostało użyte do tego 209 unikalnych tagów. Najpopularniesze tagi, w kolejności od najczęsciej użytego to python, programming, challage i puzzle.

Jedną ze stron o najnizszym rankingu jest np: http://djangosnippets.org/snippets/1314/ . Strona ta zawiera specificzne rozszerzenie dla frameworku django. Można sie spodziewać ze nie bedzie to popularna witryna. Została ona dodana przez jednego użytkownika i opisana siedmioma annotcjami.

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

3.4.1 Problemy : TODO

Potencjalne problemy zauwazone na mniejszej ilości danych: algorytm jest podatny na cykle które mogą zostać stworzone przez dużą ilość wygenerowanch 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 moze byc bezposrednio zastosowany do zebranych danych autorzy algorytmu adapted page rank zmienili strukture danych na nieskierowany graf trzydzielny$G_f = (V, E)$.

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 sie krawędziami grafuG_f: $E = \{\{u,t\},\{t,r\},\{r,u\}|(u,t,r) \in Y\}$. Wagi tych krawędzi są przydzielane w następujący sposób: każda krawędz $\{u,t\}$ ma wage $|\{r \in R : (u,t,r) \in Y\}|$, czyli jest to ilość dokumentów, którym użytkownik u nadał annotacje 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 wejsciowe:

A – prawo stochastyczna macierz sąsiedztwa grafu G_f
p – wektor preferencji
w – losowo zainicjalizowany wektor
α, β, γ – stale, gdzie: $\alpha, \beta, \gamma \in [0,1]$ i $\alpha + \beta + \gamma = 1$
do:
$w = \alpha * w + \beta * A * w + w * p$
while: do czasu kiedy wartosci wektora w zbiegna sie.

wynikiem algorytmu jest wektor w.

Dane zostały uzyskane dla paremetrów: $\alpha = 0.35, \beta = 0.65, \gamma = 0$ i pochodzą z pracy (cytat! / połączenie z bibliografia).

W algorytmie adapted page rank wektor preferencj i $p = 1$.

4.1.1 Algorytm FolkRank

Algorytm FolkRank jest wersją algorytmu adapted page rank, który 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 bedzie propagowany na reszte dokumentów, tagów i użytkowników. Można dzieki temu, ustając 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 działajacych w czasie rzeczywistym, ale nie jest użyteczny w zaprezentowanym systemie. Nie jest mozliwy do wykorzystania z powodu długiego czasu obliczania wag dokumentów.

4.1.2 Przykladowe 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 powyzszych danych ma wymiary 8×8 i wygląd:

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

	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

Zbierzność wektora została uzyskana po 22 iteracjach.

Patrząc na powyższe wyniki najwyższy ranking wśród dokumentów ma strona begence.net: została ona dodana przez 2 użytkowników i przypisane jej zostały 4 tagi. Niewiele niższy ranking ma witryna colourlovers dodana przez 2 uzytkowników i opisana 2 różnymi annotacjami. Można po tym wywnioskować ze nadanie większej ilości tagów nie ma dużego wpływu na rank strony. Za to zmniejszenie liczby użytkowników którzy tą strone dodali, ma duze: przykład strona ted.com i colourlovers.com gdzie widoczny jest dość duzy 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 przegladanie i serializowane do pliku. Dodatkowo żeby przyśpieszyć tworzenie danych korzystamy z dodatkowych tabel zawierających już wyliczone dane np: o ilosci dokumentów dodanych i opisanych tym samym tagiem przez uzytkowników.

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 zblizone do danych wymaganych w algorytmie socialpagerank wykorzystywane są te same zserializowane dane. Dokładniej macierz wygląda nastepująco:

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

gdzie tworzenie macierzy M_{UD}, M_{TD}, M_{UT} jest opisana w rozdziale opisującym algorytm social page rank.

Dodatkowo podobnie jak w algorytmie socialPageRank z powodu wielkości macierzy i ograniczen pamieciowych 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:

Zbierzność wektora uzyskano szybciej - bo już po pięciu iteracjach przy algorytmi SocialPageRank. W algorytmi Adapted Page Rank wymagało to aż 22 iteracji. Zbierzność można przyśpieszać przez zmiane 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 najwiekszej randze: behence.net i kolejnego colourlovers.com różnica dla algorytmie adapted page rank jest niewielka proporcjonalnie do wagi, a przy social page rank waga behence.net jest prawie dwa razy wieksza od poprzednika. Widoczna podobnieństwo jest dopiero dla dokumentów colourlovers.com i ted.com gdzie różnica wyników algorytmów dla jednego i dokumentu jest znaczna. Jest to prawdpodobnie spowodowane, że na wynik jednego i drugiego algorytmu wpływa mała ilość użytkowników, którzy dodali dany dokument i mała ilość tagów przypisanych temu dokumentowi.

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

Rozdział 6

Bibliografia

Paul Heymann, Georgia Koutrika, Hector Garcia-Molina. Can Social Bookmarking Improve Web Search? Sihem Amer Yaria, Michael Benedikt, Bohannon Philip. Challenges in Searching Online Communities.

Andreas Hotho, Robert Jäschke, Christoph Schmitz, Gerd Stumme. Information Retrieval in Folksonomies: Search and Ranking. Pierre Andrews, Juan Pene, Ilya Zaihrayeu. Sense Induction in Folksonomies.

Anlei Dong, Ruiqiang Zhang, Pranam Kolari, Jin Bai, Fernando Diaz, Yi Chang, Zhaohui Zheng. Time is of the Essence: Improving Recency Ranking Using Twitter Data.

Yan'an Jin, Ruixuan Li, Kunmei Wen, Xiwu Gu, Fei Xiao. Topic-based ranking in Folksonomy via probabilistic model.

Shenghau Bao, Xiaoyuan Wu, Ben Fei, Guirong Xue, Zhon Su, Yong Yu. Optimizing Web Search Using Social Annotations.

Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. An Introduction to Information Retrival.

http://pl.wikipedia.org/wiki/TFIDF