WEDT

dokumentacja końcowa projektu

Opiekun Projektu:
dr inż. Grzegorz Protaziuk

Autorzy:
Andrzej Fiedukowicz
Emil Jaworski
Daniel Pogrebniak

Treść zadania

Implementacja metody wykrywania/wskazywania synonimów

Celem zadania jest implementacja metody wykrywania/wskazywania synonimów opisanej w jednym z poniższych artykułów. Należy również przeprowadzić testy pozwalające na ocenę jakościową metody.

Opis końcowego produktu

W ramach projektu zrealizowano aplikację pozwalającą na wyszukiwanie tekstów w ich bazie na podstawie zadanych fraz wyszukiwania z uwzględnieniem synonimów owych fraz. Aplikacja przygotuje zestaw synonimów na podstawie analizy tekstów wejściowych, indeksuje ich treści a następnie umożliwia wyszukiwanie uwzględniając, przy ocenie poprawności synonimu w kontekście pewnego słowa, zarówno początkowo określoną jakość jak i preferencje wyszukujących użytkowników (częstość wyborów).

Implementacja i specyfika modułów

W tym rozdziale zaprezentowano specyfikę trzech następujących po sobie modułów z podziałem na ich elementy składowe.

1. Moduł komunikacyjny

Moduł komunikacyjny napisany jest w języku Java. Zadaniem tego modułu jest przekazywania informacji między innymi modułami projektu i dostarczanie ich do przetworzenia w określonym formacie.

Pierwszym z tego rodzaju zadań jest wczytywanie kolejno tekstów z bazy danych oraz przekazywanie ich do skryptu uruchamiającego narzędzie RASP. Program uruchamia ten skrypt w pętli dla każdego tekstu w bazie danych. Efektem działania tej pętli jest 1 plik wynikowy, który zawiera listę rzeczowników i słów z nimi sąsiadujących we wszystkich tekstach.

Powstały w ten sposób plik jest przekazywany do kolejnego skryptu (w języku Python), który wykonuje analizę otrzymanego pliku pod kątem występowania potencjalnych synonimów. Dokładny opis tego skryptu znajduje się w dalszej części tego dokumentu.

Ostatnim krokiem programu jest wczytanie pliku wyjściowego ze skryptu w języku Python, sparsowanie tego pliku i umieszczenie potencjalnych synonimów w bazie danych. Wykonienie tego kroku jest konieczne ze względu na fakt, że wyszukiwarka tekstów opiera się na liście potencjalnych synonimów z bazy danych.

2. Przetworzenie tekstu

W oparciu o literaturę [1,3] wypracowano następujące mechanizmy przetwarzania tekstu w celu wstępnej selekcji synonimów odpowiadających zadanej liście rzeczowników.

Składowe narzędzia

Przetwaranie tekstów na potrzeby wyszukiwarki wykonywane jest przy pomocy dwóch skryptów:

- *start.sh* - uruchomienie skryptów RASP i dodanie wyniku do pliku
- *getText.py* - przetwarza plikt wynikowy skryptów RASP, analizuje otrzymane dane i wykonuje bazę synonimów wraz z ich oceną.

Analiza tekstu

Analiza tekstu wykonywana jest przy pomocy gotowego narzędzia RASP, pracującego z językiem angielskim. W wyniku przetwarzania tekstu otrzymujemy podział na zdania, relacje między słowami oraz ich częsci mowy. Wynik jest następnie zapisywany do pliku w formacie wykorzystywanym przez RASP.

Kolejnym krokiem jest uruchomienie skryptu wykonanego w języku Python. Skrypt ten wczytuje dane wygenerowane przez RASP i parsuje je w celu wczytania do struktur danych umożliwiających przetworzenie.

Gromadzenie danych o słowach kluczowych

Z danych otrzymanych ze skryptu RASP odczytujemy informacje takie jak część mowy oraz połączenie słów w zdaniu. Dzięki temu możliwe jest wygenerowanie listy rzeczowników wraz ze słowami opisującymi je. W skrypcie przetrzymujemy listę wszystkich rzeczowników i słów bazowych oraz odpowiadające im listy słów "opisujących" je - czyli takie które znajdują się w sąsiedztwie rzeczowników. Przykładem słowa opisującego "melon" jest słowo "żółty" w zdaniu "Melon jest żółty".

Wyszukiwanie synonimów

Wyszukanie synonimów wykonywane jest przez porównanie liczby takich samych słów opisujących dwa różne rzeczowniki. Dla każdego porównania obliczana jest ocena mająca na celu określenie oceny synonimów:

$$Q_{base}(w,s) = \frac{\sum\limits_{x \in X_{w,s}} (Count_w(x) \times Count_s(x))}{\sum\limits_{x \in X_w} Count_w(x) \times \sum\limits_{x \in X_s} Count_s(w)} \times \frac{|X_{w,s}|}{|X_w \cup X_s|}$$

$Q_{base}(w,s)$ – Base quality of synonym s for word w

$X_{x_1,x_2,\ldots,x_n}$ – Set of words defined as $\bigcap\limits_{i=1}^{n} C_{x_i}$ where C_{x_i} is set of words connected to x_i literal.

$Count_l(x)$ – Number of occurences of word x near the literal l

Wynik

Wynikiem działania skryptu jest lista słów bazowych i odpowiadających im potencjalnych synonimów wraz z oceną. Lista synonimów została ograniczona do trzech dla każdego słowa.

Testy

Testy algorytmu przeprowadzone zostały dla książki składającej się z **12468 linii tekstu**. Przetworzenie książki przy pomocy skryptów RASP wykonywały się: **4 minuty 11 sekund**
Skrypt analizujący wykonywał się: **3 minuty i 14 sekund**

Poniżej przedstawione zostały wynik dla 9 słów bazowych w formacie:
`SłowoBazowe:synonim1(ocena1),synonim2(ocena2)...`

```
car:front(0.0209385504474),left(0.016474331583),right(0.0160281247898)
cat:tilt(0.0115838716863),spoor(0.00796152925214),animal(0.00761746821448)
hair:course(0.00867582788663),neck(0.00744265617484),face(0.00727364289137)
guard:raincoat(0.00783616292045),desk(0.00699636661486),temptation(0.00690785990364)
orange:intensity(0.226001511716),root(0.0825840825841),chauffe(0.0555555555556)
position:course(0.0167119101931),ren(0.01440334666),bunker(0.014253272399)
ring:course(0.0123736054909),shadow(0.00787394833409),hallway(0.00656059718323)
match:course(0.0166083075174),kind(0.0103398481218),sight(0.0094634515844)
panel:course(0.00966003750056),end(0.00874017905067),shadow(0.00846340787863)
```

Zaimplementowany algorytm nie zwraca synonimów dla danego tekstu jednak wyniki przedstawiają słowa “zbliżone” do słowa bazowego. Dobrymi przykładami są np:

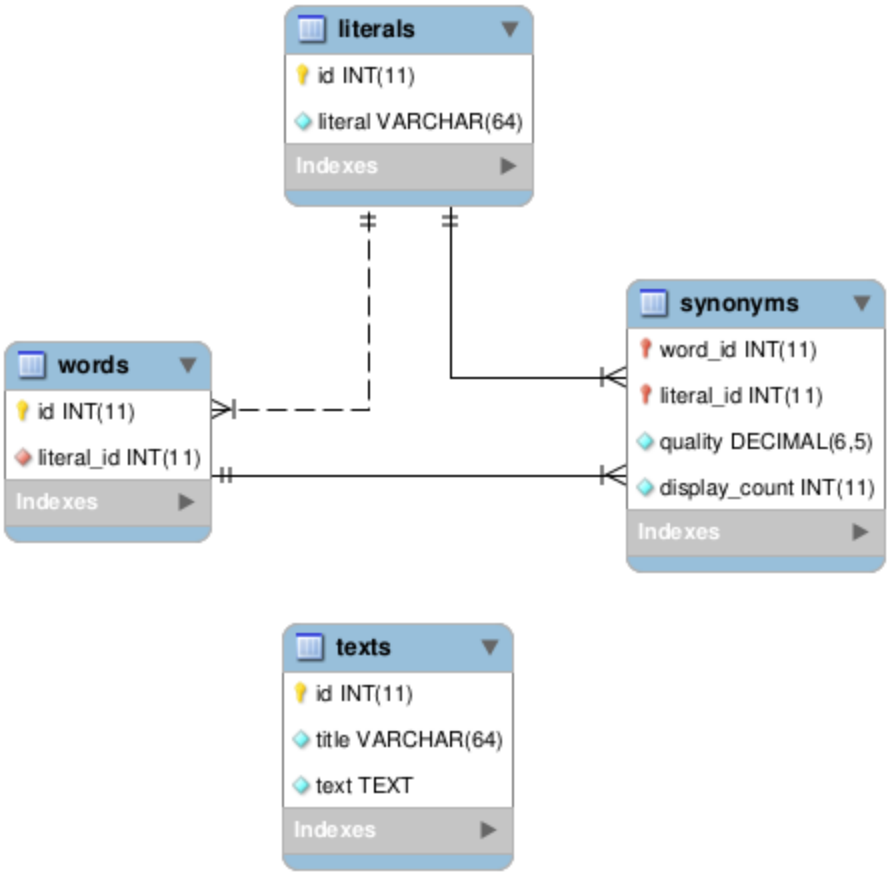
- **cat:** animal
- **hair:** neck - face

Wyniki te zostaną wprowadzone do “wyszukiwarki”, która wzmocni otrzymane oceny i dzięki temu możliwe będzie otrzymanie prawdziwych synonimów.

3. Adaptacja bazy synonimów

Wygenerowana w pierwszym kroku baza synonimów, zgodna z ustalonym w ramach dokumentacji wstępnej strukturą trafia do modułu wyszukiwarki, który zajmuje się indeksowaniem treści i jej prezentacją na podstawie zapytań i zachowań użytkowników w ramach konkretnych wyników.

Jako, że ze względów implementacyjnych struktura bazy danych uległa drobnej modyfikacji względem wstępnej wersji, poniżej przedstawiono jej ostateczną strukturę w formie diagramu ER.



Indeksowanie

Pierwszym zadaniem modułu wyszukiwarki było zaindeksowanie tekstów dostarczonych do systemu. Indeksowanie, zgodnie z przyjętymi we wstępnej dokumentacji założeniami, zostaje wykonane za pomocą fragmentu zestawu bibliotek Zend Framework - biblioteką **Zend Lucene**.

Do biblioteki tej przekazywane są w celu zaindeksowania kolejno wszystkie teksty w bazie danych i zostaje utworzony indeks na podstawie ich identyfikatorów, tytułu i tekstu. W indeksie przechowywane są pierwsze dwie z tych danych a także pierwsze N = 255 znaków tekstu, by umożliwić fragmentaryczne wyświetlanie treści wyników wyszukiwania bez konieczności wykonywania kosztownych zapytań do bazy danych.

W trakcie indeksowania tworzone są też sztuczne synonimy słów o takich samych literałach jak słowo oryginał. Zabieg ten, służy uogólnieniu algorytmów przetwarzania, upraszcza implementację i pozwala na uzyskanie lepszych rezultatów i wystąpienie w wyniku adaptacji bazy sytuacji, w której dla wyszukiwania pewnego słowa, cenniejszym dla użytkowników wyszukiwarki są wyniki dla jego synonimu - mechanizm podobny do mechanizmu *"did you mean"* znanego z google.

Cały proces indeksowania wykonywany jest w ramach skryptu *web_ui/create_index.php.*

Adaptacja jakości synonimów

Najistotniejszym elementem działania wyszukiwarki jest adaptacja jakości synonimów na podstawie akcji użytkownika. W tym przypadku akcjami badanymi przez wyszukiwarkę są wybory konkretnych wyników wyszukiwania (przejścia na stronę z konkretnym tekstem).

Adaptacja bazy synonimów odbywa się więc w oparciu o zapisywaną w bazie ilość wybrań danego synonimu przy wyszukiwaniach dla konkretnego słowa bazowego. Wartość ta przechowywana jest w kolumnie *display_count* tabeli *synonyms* bazy danych.

W oparciu o literaturę [2], wyprowadzono następującą formułę opisującą jakość synonimu:

$$Q(w,s) = Q_{base}(w,s) \times Q_{click}(w,s)$$

$$Q_{click}(w,s) = \frac{Clicks(w,s) + C}{\frac{1}{|S_w|}\sum_{\sigma \in S_w}(Clicks(w,\sigma) + C)}$$

$Q(w,s) -$ Quality of synonym s for word w
$Q_{base}(w,s) \in [0,1] -$ Base quality of synonym s for word w from preprocessing
$Clicks(w,s)$ – Number of times user has chosen result based on synononym s while searching for w
$S_w -$ Set of synonyms connected with word w
$C -$ constant value used to get smooth startup and to get rid of zeros problem

Interfejs użykownika

W celu wygodnego dostępu do utworzonych zasobów został stworzony interfejs użytkownika oparty o technologie HTML i CSS przy wykorzystaniu języka PHP do obróki danych po stronie serwera i umieszczania ich w obrębie dokumentów dla użytkownika.

Interfejs aplikacji dostępny jest w ramach pliku *index.php*. Składa się on z następujących, standardowych dla współczesnych wyszukiwarek elementów:

- Paska wyszukiwania
- Wyników wyszukiwania

- Podstron dla konkretnych wyników

Wyszukiwarka ma na celu podawać najlepsze rezultaty dla wyszukiwanej frazy i w związku z tym, wyniki prezentowane są w kolejności wynikającej z wyznaczonej dynamicznie jakości synonimów.

Ze względów informacyjnych, do wyników wyszukiwania dołączono informację z jakiego powodu dany wynik znalazł się na liście, co opisują zdania typu *"did you mean"*. Z kolei ze względów edukacyjnych i testowych, zdania te pojawiają się przy każdym wyniku wyszukiwania, a ponadto wyświetlany jest także wyznaczony przez system procent zgodności danego synonimu z wyszukiwanymi frazami (dla zgodności z wieloma frazami, synonim zawsze traktowany jest jako pochodzący od tej frazy, której system daje największą zgodność).

Warto zwrócić uwagę, że by ułatwić wyszukiwanie i zwiększyć praktyczność narzędzia, zostało zastosowane wyszukiwanie przybliżone w ramach indeksu (funkcjonalność dostępna w bibliotece Zend Lucene). Pozwala ono na wyszukiwanie odmian słów, słów obarczonych drobnymi błędami itp. Jest to mechanizm typowy dla współczesnych wyszukiwarek.

Poniżej załączono zrzut ekranu z wyszukiwarki po wyszukaniu tekstów (teksty w języku polskim były wykorzystane tylko do celów testów wyszukiwarki, wstępna obróbka Polskich tekstów nie jest możliwa).

Środowisko pracy

W celu uruchomienia kompletnej aplikacji wymagane jest środowisko wyposażone w następujące narzędzia:

- Bash
- Interpreter Języka Pascal
- Java Runtime Environment
- RASP
- Interpreter Języka PHP
- Baza danych MySQL
- Zend Framework dla języka PHP
- Serwer HTTP
- Przeglądarka WWW

Instrukcja użytkownia

W celu uruchomienia aplikacji (przy zainstalowanym, kompletnym środowisku jw.) należy wykonać następujące kroki:

1. Umieścić wewnątrz bazy danych teksty wymagające przetworzenia a także stworzyć listę rzeczowników dla których wyszukiwane będą słowa. Dla utworzenia struktury tabel, należy użyć skryptu *db/db.sql.*
2. Uruchomić aplikację w Javie w celu przetworzenia tekstów - wcześniej konfigurując w jej ramach dostęp do bazy danych
3. Umieścić w publicznym folderze pliki z katalogu *web_ui* - wcześniej konfigurując w jej ramach dostęp do bazy danych
4. Uruchomić skrypt *create_index.php*
5. Wykorzystywać plik *index.php* do wyszukiwania elementów w bazie z wykorzystaniem synonimów.

Informacje dodatkowe

- Do poprawnego działania program wymaga aby dany rzeczownik i jego synomin wystąpiły kilkukrotnie w sąsiedztwie takich samych słów. Nie każdy test posiada taką własność, dlatego dla niektórych tekstów otrzymane wyniki mogą odbiegać od wyników oczekiwanych.
- Przetwarzanie tekstów jak i tworzenie indeksów może być procesem długotrwałym.
- Teksty jak i rzeczowniki powinny być sporządzone w języku angielskim, ze względu na fakt, że tylko ten język jest obsługiwany w ramach narzędzia RASP.

Bibliografia

[1] Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama: "Selection of Effectiv Contextual Information for Automatic Synonym Acquisition", In Proceedings of COLING/ACL 2006

[2] Kaushik Chakrabarti, Surajit Chaudhuri, Tao Cheng, Dong Xin: "A Framework for Robust Discovery of Entity Synonyms" in KDD '12 Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining

[3] P. Turney: "Mining the web for synonyms: PMI-IR versus LSA on TOEFL", ECML 2001

Załączniki

- Repozytorium kodu pozwalające na dostęp do gotowej implementacji:
    - https://github.com/fiedukow/WEDThesaurus