Wyszukiwarka postów na Twitterze

Piotr Jarosik, Michał Karpiuk, Marcin Wlazły

Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska
P.Jarosik@stud.elka.pw.edu.pl
M.Karpiuk.1@stud.elka.pw.edu.pl
M.Wlazly@stud.elka.pw.edu.pl

Streszczenie Niniejszy praca stanowi dokumentację projektu zrealizowanego w ramach przedmiotu *Wstęp do Eksploracji Danych Tekstowych*. Celem było zaimplementowanie prostej wyszukiwarki tekstowej. Zrealizowany system zawiera podstawowe mechanizmy odporności na błędy w zapytaniach i treści postów oraz uwzględnia niejednoznaczności słów.

Słowa kluczowe: wyszukiwarka postów, twitter, elasticsearch, spring, spring-data, WordNet

W ramach projektu zrealizowane została wyszukiwarka postów Twittera z wykorzystaniem indeksera tekstowego opartego na silniku Lucene. Dokumentacja składa się z

1 Korpus tekstowy

Korpus tekstowy wykorzystany w ramach projektu składa się z dwóch zbiorów wiadomości tweeter [1]:

- zawierających w treści wystąpienia *apple*;
- otagowane jako `#iranelection`.

Uwzględnione zostały wiadomości tylko w języku angielskim.

Każda wiadomość składa się z kilkunastu pól oddzielonych znakiem `0x09` (*ASCII horizontal tab*). W ramach projektu uwzględnione zostały:

- data utworzenia wiadomości;
- nazwa użytkownika oraz jego awatar;
- kod kraju autora wiadomości;
- treść wiadomości.

Trzy pierwsze z wymienionych atrybutów przechowywane są wyłączenie w celu prezentacji wyników wyszukiwania użytkownikowi. Pola te nie są również uwzględniane we wstępnej analizie przed indeksowaniem. Treść wiadomości jest natomiast indeksowana w postaci wynikowej przygotowanej przez analizator systemu.

[1] `http://an.kaist.ac.kr/~haewoon/release/twitter_tweets/`

2 Wykorzystane narzędzia

2.1 Elasticsearch

W ramach projektu zastosowano indekser **elasticsearch**. Działa on jako całkowicie niezależna usługa w systemie, nie ma więc konieczności jego uruchamiania w serwerze aplikacyjnym (jak jest to np. w Solr).

Podstawę działania tego narzędzia stanowi biblioteka Lucene. Instancja elasticsearch nazywana jest **węzłem** i może ona zawierać jeden lub wiele indeksów. Jeden indeks elasticsearch składa się z:

- jednego głównego fragmentu indeksu (ang. *primary shard*);
- jednej lub wielu replik głównego fragmentu indeksu (ang. *replica shard*).

Każdy fragment indeksu elasticsearch jest **jednym indeksem Lucene**. Dzięki temu możliwa jest realizacja systemu niezawodnościowego - w przypadku awarii głównego fragmentu indeksu automatycznie wybierana jest jedna z jego replik.

Komunikacja między indekserem a aplikacją odbywa się poprzez interfejs RESTowy, za pomocą komunikatów zapisanych w formacie JSON (stanowi on DSL [2] elasticsearch'a). W szczególności, dokumenty zapisywane do indeksu muszą być obiektami zgodnymi z tą notacją.

Elasticsearch posiada wiele innowacyjnych rozwiązań, wynikających z bardzo dużej liczby wtyczek (plugin's) przygotowywanych przez deweloperów z całego świata. Szczególną ciekawą funkcją tego narzędzia są perkolacje - możliwe jest wykonanie odwróconego zapytania. W indeksie można zapisać różne frazy wyszukiwania, a następnie przekazać do elasticsearch dokument, w celu znalezienia wszystkich zapisanych fraz, które są relewantne dla wprowadzonego dokumentu. W ten sposób można np. w prosty sposób wykonać klasyfikację dokumentów wprowadzanych przez użytkowników systemu.

2.2 Spring framework

W ramach tego projektu opracowana została aplikacja webowa z wykorzystaniem technologii JEE. Zastosowano szkielet aplikacyjny Spring Framework, w szczególności:

- **w warstwie prezentacji** – Spring Web MVC oraz Apache Tiles, widoki użytkownika generowane są z plików JSP;
- **w warstwie dostępu do danych (przechowywanych w indekserze)** – narzędzie spring-data, które umożliwia przygotowanie abstrakcji repozytorium danych. W Spring Data nie ma potrzeby pisania własnych zapytań (np. obiektów w formacie JSON do indeksera) – zapytania są przygotowywane automatycznie **wywnioskowywane** z nazw metod interfejsu repozytorium. Spring-data może być również zastosowany m.in. w relacyjnych oraz grafowych bazach danych. Istotną wadą tego narzędzia jest intensywne używanie refleksji języka Java – powoduje to znaczny spadek wydajności systemu (w przyszłości ma być użyty mechanizm `MethodHandle` dostępny dla java 1.7).

[2] język dziedzinowy, ang. Domain Specific Language

3 Indeksowanie danych wejściowych

W ramach projektu wykorzystany został korpus tekstowy opisany w rozdziale 1.

Dane, przed zaindeksowaniem, zostały oczyszczone z nieistotnych elementów (z punktu widzenia realizacji wyszukiwania), takich jak odnośniki do zdjęć czy zewnętrznych stron (tak, aby nie były uwzględniane w dalszej analizie). Ponadto poddane one zostały działaniu elementów standardowego analizatora Lucene, tj:

- standardowy tokenizer implementujący algorytm segmentacji tekstu Unicode (Unicode Text Segmentation algorithm) uzupełniony o ignorowanie adresów url oraz email (*uax_url_email tokenizer*).
- filtr *stop words* dla języka angielskiego;
- filtr zmieniający litery na małe;
- filtr normalizujący tokeny.

Dodatkowo, zastosowane zostały następujące filtry tokenów:

- stemmer dla języka angielskiego;
- filtr do przygotowywania bigramów z danych wejściowych (shingle filter z biblioteki Lucene);
- filtr uwzględniający synonimy wyrazów.

Aby skorzystać z tego ostatniego, konieczne było przygotowanie pliku z synonimami w formacie akceptowalnym przez Solr. Niezbędne było więc skorzystanie ze słownika języka angielskiego.

W ramach projektu wykorzystane zostały dane z Wordnet-u [3], który jest leksykalną bazą danych języka Angielskiego. Z punktu widzenia projektu szczególnie interesujące były:

- synonimy (niezbędne dla wskazanego wcześniej filtru analizatora);
- dane do sprawdzania treści zapytań.

Elasticsearch umożliwia zastosowanie informacji bezpośrednio z Wordnet-u, jednak w związku z błędem występującym w indekserze (nie wszystkie postacie są obecnie akceptowane przez elasticsearch) dane te zostały skonwertowane do postaci zgodnej ze składnią Solr.

4 Wyszukiwanie informacji w indeksach

Zarówno indeksowane dokumenty, jak i zapytania zostały poddane działaniu tych samych analizatorów Lucene. Do obsługi błędów w zapytaniach wykorzystano suggester termów [4], który na podstawie podanej frazy oraz na podstawie zaindeksowanych postów podaje sugestie dla prawdopodobnie błędnych wyrazów, tj. słowa z dokumentów przechowywanych w indeksie najbliższe do podanego ciągu wyrazów. Np. dla frazy

[3] `http://wordnet.princeton.edu/`

[4] nowa funkcjonalność elasticsearch'a, ciągle jeszcze w fazie beta: `http://www.elasticsearch.org/guide/reference/api/search/suggest/`

“spatła kedytu” użytkownik otrzyma informację, iż prawdopodobnie chodziło mu o frazę “spłata kredytu”.

Suggester termów wykorzystany został również przed zaindeksowaniem postów (w sytuacjach gdy było to konieczne). Dla każdego wyrazu, który jest z prawdopodobnie błędny, uwzględniona jest propozycja od suggester'a. To, czy dane słowo zostało uznane za błędne, wyznacza wartość progowa odległości słowa z zapytania do proponowanego przez suggester wyrazu. Parametr ten został ustalony w trakcie eksperymentowania z wyszukiwaniem zaindeksowanych danych i wynosi 0.5.

Użytkownik końcowy ma możliwość wpisania frazy do wyszukiwania w specjalnie przygotowanym do tego formularzu. W odpowiedzi wyświetlana jest lista pasujących do zapytania postów. Wyniki wyszukiwania jest stronicowana, tj. na jednej stronie będzie można wyświetlić pewną określoną, maksymalną liczbę wyników. Do stronicowania została wykorzystana technologia AJAX, dzięki czemu posty wczytują się dynamicznie, a użytkownik nie musi za każdym razem przeładowywać strony. Każdy wynik wyszukiwania zawiera: treść postu, awatar oraz nazwę autora. Dane są wyświetlane rosnąco wg wagi `tf-idf` .

5 Metody testowania

W ramach projektu zaindeksowano 85252 wiadomości, zarówno z korpusu tekstowego (przedstawionego w 1) jak i portalu `twitter.com`. Na danych testowych przeprowadzone zostały testy jakości wyszukiwania. Wykonane testy zostały zaprezentowane w kolejnych podrozdziałach:

5.1 Testy jednostkowe

Testy jednostkowe zostały przeprowadzone z wykorzystaniem framework'a TestNG. Jest to ciekawy szkielet aplikacyjny stanowiący alternatywę dla biblioteki JUnit. Umożliwia on elastyczną konfigurację testów, wykorzystuje się go przy pomocy adnotacji Java.

5.2 Testy jakościowe

W ostateczności nie zdecydowaliśmy się na próby obliczania miar poprawności takich jak precyzja czy zupełność. Byłoby to wyjątkowo trudnym zadaniem, biorąc pod uwagę fakt, że wyniki testów w bardzo dużym stopniu zależałyby od skomplikowania zapytania, co również jest kwestią subiektywną. Takie wyniki były by raczej mało precyzyjne. Ponadto aby obliczyć wskaźnik zupełności niezbędna jest wiedza o wszystkich poprawnych przypadkach postów w korpusie, w kontekście danego zapytania. Taką wiedzę można zdobyć uważnie przeglądając wszystkie posty, jednak nakład pracy potrzebny na jej zdobycie jest zdecydowanie nieakceptowalny. Z drugiej strony ograniczenie zbioru testowego do wielkości umożliwiającej zapanowanie nad testami bardzo zmniejszyłoby dokładność.

W związku z tym przeprowadzone zostały testy dla wybranych zapytań.

Testy poprawności wyszukiwania

1. Wyszukiwanie proste, fraza wejściowa: *apple juice*.

   Liczba wszystkich wyników: 2907.
   Najbardziej relewantny wynik (tf-idf = 3.0314815):
   Strawberry Kiwi Apple juice and Campbell's chunky soup for breakfast! Yummy!

5.3 Testy wydajnościowe

Testy wydajnościowe miały zostać przeprowadzone za pomocą frameworka The Grinder, który umożliwia łatwe i szybkie tworzenie testów wydajnościowych, w tym również na oszacowanie wydajności systemu w przypadku jego użycia przez wielu użytkowników jednocześnie. Ze względu na brak miarodajnego środowiska deweloperskiego (odpowiadającego środowisku produkcyjnemu) nie udało się przeprowadzić testów wydajnościowych.

Literatura