Pracownia z wyszukiwania informacji

Prowadzący: dr hab. Tomasz Jurdziński

Paweł Murias, Michał Rychlik

Wrocław, dnia 17 maja 2011 r.

1 Instalacja

Na płycie załączone są programy wyszukujące w formacie binarnym: tokenize.bin i search.bin. Należy je uruchomić w katalogu zawierającym morfologik. Aby utworzyć indeksy należy odpalić *tokenize.bin sciezkadowikipedii* oraz *tokenize.bin sciezkadowikipedii* 1, aby utworzyć indeks używający stemmingu.

Aby uruchomić wyszukiwarkę ze źródeł wymagana jest aktualna instalacja perl oraz instalacja wykorzystanych modułów.

$curl\ -L\ http://xrl.us/perlbrewinstall\ |\ bash$
$echo\ 'source\ /perl5/perlbrew/etc/bashrc'\ >>\ /.bashrc$
$//otworzyc\ nowa\ powloke$
$perlbrew\ install\ perl-5.12.3$
$curl\ -L\ http://cpanmin.us\ |\ perl\ -\ App::cpanminus$
$cpanm\ Term::ProgressBar::Quiet$
$cpanm\ Inline::C$
$cpanm\ Dir::Self$
$cpanm\ utf8::all$
$cpanm\ List::MoreUtil$

2 Instrukcja użytkownika

Aplikacja działa w dwóch trybach:

- wsadowym
- interaktywnym

Podanie opcji –stemmer powoduje użycie indeksu ze stemmingiem a –compressed z kompresją. Można łaczyć obie opcje. W trybie wsadowym aplikacja oczekuje na dwa argumenty wywołania programu. Pierwszym ma być plik z zapytania w formacie wynikającym z treści zadania, a drugim nazwa pliku w którym mają znaleźć się wyniki.

W trybie interaktywnym, który uruchamiany jest jeśli użytkownik nie poda argumentów, które powodują uruchomienie trybu wsadowego aplikacja wyświetla symbol zachęty (prompt), po którego pojawieniu się użytkownik może wpisywać zapytania w formacie określonym w treści zadania.

3 Opis użytych algorytmów

Indeks to zbiór pozycyjnych list postingowych utrzymywany w binarnym pliku na dysku twardym. W pliku tym dane przechowywane są w następujący sposób:

- Pierwszą wartością jest liczba termów w słowniku.
- Potem następuje blok liczb o długości, odpowiadającej liczbie termów, które oznaczają miejsce w pliku (offset), w którym powniśmy szukać listy postingowej dla danego termu. Tak więc w i-tej linni tej sekcji znajduje się "adres" w dalszej części pliku, pod którym można znaleźć listę postingową i-tego termu. Dzięki temu dostęp do interesującego fragmentu pliku osiągany jest w czasie stałym.
- W pozostałej części pliku zaczynając się pod wspomnianymi wyżej offsetami znajdują się listy postingowe poszczególnych termów. Offset oznacza początek listy danego termu, jej koniec jest określany albo przez offset dla kolejnego termu albo przez koniec pliku w przypadku ostatniego termu. Lista postingowa termu składa się z jednej lub większej ilości sekcji postaci: (docID, posSize, positions), gdzie: docID to id dokumentu w którym występuje dany term, posSize to ilość miejsc, w których dany term występuje w dokumencie o id równym docID, a postings to lista pozycji wystąpień termu w dokumencie o długości posSize.

Z powodu wielkości danych do zaindeksowania oraz ograniczonej ilości pamięci RAM, na maszynie testowej, indeks jest tworzony w częściach. Każda z części jest wynikiem działania tokenizatora, który przegląda dokumenty do zaindeksowania i tworzy listę trójek (tokenID, docID, position), które następnie są sortowane leksykograficznie, co pozwala w łatwy sposób stworzyć pliki o podanym powyżej formacie. Następnie zewnętrzny skrypt dokonuje scalenia części indeksu w jeden plik wynikowy (dla danych z wikipedii w procesie tym są tworzone 24 pliki częściowe, a ostateczny, scalony plik indeksu ma ok. 1GB).

Istnieje również możliwość wyszukiwania w indeksie w formie skompresowanej. Kompresja dotyczy tylko list postingowych. Ponieważ zależy nam na ważnej własności dostępuj w czasie stałym do odpowiedniej sekcji pliku na podstawie offsetu, o którym wiemy, że znajduje się na i-tej pozycji w pierwszej części pliku. Nie da się tego osiągnąć za pomocą kompresji, którą stosujemy w naszym rozwiązaniu.

Algorytm kompresji postępuje jak następuje:

- Chcąc skompresować liczbę n zapisujemy ją w systemie o podstawie 128.
- Każda z liczb występujących w tym zapisie może być zapisana na 7 bitach.

- Liczby, które nie są ostatnimi w reprezentacji, jeden nadmiarowy bit (z 8 dostępnych) przy implementacji wykorzystującej typ unsigned char, mają ustawiony na 0. Ostatnia liczba ma go ustawionego na 1, co ma sygnalizować koniec zapisu liczby n.
- Ciąg liczby (typu integer) zapisujemy jako jeden długi napis będący sklejeniem reprezentacji opisanych powyżej.
- Dodatkowo, ponieważ pozycje w dokumencie mogą mieć stosunkowo wysokie numery. Pamiętamy jedynie pierwszy numer pozycji w pełnej formie, a pozostałe zastępujemy różnicami między daną pozycją a poprzednią, licząc przy tym, że uzyskane liczby będą o wiele krótsze od orginalnych.

Lematyzacja jest osiągana przez pobieranie form bazowych z morfologika. Stemming został zaimplementowany przy pomocy następującego algorytmu:

- Na podstawie morfologika tworzymy listę 50 najpopularniejszych końcówek słow.
- Za końcówke słowa rozumiemy ten litery z końca słowa które się różnią od formy bazowej.
- Robiąc stemming odcinamy prefiks słowa "nie" lub "naj" oraz najpopularniejszą pasującą końcówkę z listy.

4 Implementacja

Aplikacja została napisana w dwóch językach programowania: Perlu oraz C.
Perl został wykorzystany głównie w procesie tokenizacji, obsługi wejścia, wyjścia (interfejsu użytkownika) oraz do testów. W C zostały napisane części programu, na których wydajności najbardziej zależało autorom (tzn. tworzenie indeksu, scalalnie plików pośrednich, wyszukiwanie).

W implementacji wykorzystano następujące biblioteki:

- Term::ProgressBar Pasek postępu przy tworzeniu indeksu.
- Inline::C Korzystanie z kodu C z poziomu perla.
- Storable Zapisywania prostych struktur perlowych na dysku.
- Test::More,File::Temp,Test::File::Contents Biblioteki używane w testach jednostkowych.

5 Wyniki testów

Proste testy poprawnościowe (na małych danych), znajdują się w katalogu t i zostały napisane w perlu. Sprawdzają one wyniki zapytań o termy, które znajdują się w słowniku, nie ma ich tam, a także zapytania boolowskie oraz frazowe. Czas wykonania (podania odpowiedzi) dla tych testów był pomijalnie mały, a ich celem było jedynie sprawdzenie poprawnośći zastosowanych algorytmów, możliwe do zweryfikowania na podstawie danych wejściowych.

Następnie wykonano szereg testów na zapytaniach podanych w treści zadania. Poniżej znajduje się zestawienie czasów, po których były dostępne wyniki.

Test(Metoda)	Czas oczekiwania na wynik
Pytania koniunkcyjne (brak kompresji, brak stemmingu)	2m 12s
Pytania koniunkcyjne (brak kompresji, stemming)	1m 24s
Pytania koniunkcyjne (kompresja, brak stemming)	10m 36s
Pytania koniunkcyjne (kompresja, stemming)	8m 23s
Pytania AND,OR (brak kompresji, brak stemmingu)	3m 17s
Pytania AND,OR (brak kompresji, stemming)	1m 48s
Pytania AND,OR (kompresja, brak stemmingu)	17m 15s
Pytania AND,OR (kompresja, stemming)	14m 14s
Zapytania frazowe (brak kompresji, brak stemmingu)	35s
Zapytania frazowe (brak kompresji, stemming)	46s
Zapytania frazowe (kompresja, brak stemmingu)	4m 25s
Zapytania frazowe (kompresja, stemming)	4m 13s

Ponadto odnotowano następujące czasy, poszczególnych części procesu rozruchu.

Metoda tworzenia indeksu	Czas trwania
Tokenizacja,tworzenie plików częściowych, scalanie (stemming)	27m 36s
Tokenizacja,tworzenie plików częściowych, scalanie (brak stemmingu)	21m 23s