MECHANIKA
CZASOPISMO TECHNICZNE
TECHNICAL TRANSACTIONS
MECHANICS
WYDAWNICTWO POLITECHNIKI KRAKOWSKIEJ
9-M/2012
ZESZYT 26
ROK 109
ISSUE 26
YEAR 109

ARTUR NIEWIAROWSKI*

OPTYMALIZACJA SCHEMATU WAŻENIA TERMINÓW DLA MODELU WEKTOROWEGO

TERM FREQUENCY OPTIMIZATION FOR THE VECTOR SPACE MODEL

Streszczenie

Artykuł opisuje wybrane metody ważenia terminów dla modelu wektorowego dokumentów tekstowych oraz wybrane metody wyznaczania podobieństw. Dodatkowo, dla zwiększenia dokładności analizy danych, zaimplementowano w procesie ważenia algorytm miary podobieństwa ciągów oparty na odległości Levenshteina. W celu przyspieszenia komparacji danych użyto technologii obliczeń równoległych.

Słowa kluczowe: data mining, text mining, obliczenia równoległe, grupowanie dokumentów

Abstract

Article describes selected terms weighted methods for the vector space model of text documents and selected methods of determine similarities. To improve accuracy of data analysis was implemented algorithm which calculates similarity measure between two strings, based on Levenshtein distance. For accelerate data comparison was used parallel computing technology.

Keywords: data mining, text mining, parallel computing, aggregate documents

* Mgr inż. Artur Niewiarowski, Instytut Informatyki, Wydział Fizyki, Matematyki i Informatyki, Politechnika Krakowska.

1. Aspekt grupowania dokumentów w analizie poprawności gramatycznej i ortograficznej

Większość mechanizmów analizujących poprawność zdań pod względem ortograficznym czy też gramatycznych opiera się na zaimportowanej wcześniej bazie tekstów – tzw. korpusie. Poprawność, ale również szybkość analizy zależy od doboru właściwych tekstów, jak również metod analizy i modeli reprezentacji danych. Istnieje możliwość stworzenia interfejsu mechanizmu, który umożliwiałby wstępne ustawienie dziedzinowości analizowanych danych. Niestety wielkość zasobów, jak również szczegółowość danych tekstowych w danej dziedzinie może znacząco spowolnić proces mechanizmu. W tym celu zastosowanie mają algorytmy grupowania dokumentów tekstowych (fragmentów tekstów) umożliwiające wstępne oszacowanie (tzn. przyporządkowanie) badanego tekstu do zasobów.

Istnieje wiele metod grupowania dokumentów, do najważniejszych należą metody: płaskie, hierarchiczne, grafowe, oparte na gęstości itd. Metody określają sposób segregacji danych i ich ostateczną reprezentację jako wynik analizy. Dla metody należy określić model danych umożliwiający reprezentację danych tekstowych (zdań oraz całych dokumentów), jak również konkretne operacje oszacowania stopnia relacyjności (podobieństwa) pomiędzy dokumentami. Do najczęściej stosowanych modeli należą: wektorowy, grafowy i przestrzeni metrycznej.

W publikacji przedstawiony zostanie najpopularniejszy model reprezentacji danych, tj. model wektorowy. Komparacja terminów odbywa się z użyciem algorytmu miary podobieństwa ciągów opartego na odległości Levenshteina w celu dokładniejszego oszacowania ich podobieństwa i wyliczeniu odpowiedniej wagi. Wprowadzenie w każdy mechanizm dodatkowych elementów obliczeniowych (w tym przypadku miary podobieństwa ciągów) powoduje obniżenie czasu wykonania obliczeń. Dlatego w publikacji zaprezentowany został dodatkowo algorytm umożliwiający zrównoleglenie analizy dokumentów tekstowych. Optymalizacja procesu ważenia terminów polega na zwiększeniu dokładności oszacowania wag pomiędzy terminami bez użycia dodatkowych słowników (np. wyrazów bliskoznacznych), jak również przyspieszeniu głównego algorytmu komparacji dokumentów poprzez zastosowanie technologii obliczeń równoległych.

2. Schematy ważenia w modelu wektorowym

Model wektorowy (ang. *Vector Space Model*) jest najpopularniejszym modelem reprezentacji dokumentów tekstowych. Dokument (fragment dokumentu) jest przedstawiony jako *n*-wymiarowy wektor cech:

$$\boldsymbol{d}_j = (\omega_{j,i}, \omega_{j,i+1}, ..., \omega_{j,n-1}, \omega_{j,n}),\ i \in \langle 1, n \rangle \tag{1}$$

gdzie:

- $\boldsymbol{d}$ – wektor reprezentujący dokument (fragment dokumentu),
- j – identyfikator dokumentu,
- i – liczba naturalna oznaczająca kolejne słowa lub grupy słów określające termin,
- n – liczba terminów w j-tym dokumencie,
- $\omega_{j,i}$ – cecha, liczba rzeczywista będąca wartością znaczeniową terminu.

Ważenie to proces, który nadaje każdemu terminowi (tj. słowo, grupa słów, ang. *term*) wartość znaczeniową, najczęściej jest nią liczba wystąpień w dokumencie. Tego typu schemat określany jest w języku angielskim jako *term frequency*: $tf_{t,d}$ (gdzie: t – termin, d – dokument). Poważnym problemem tego typu modelu jest brak rozróżniania istotności poszczególnych terminów charakteryzujących zagadnienia (dziedzinę) zawarte w dokumencie. Innym rozwiązaniem jest oszacowanie danego wystąpienia w badanych dokumentach i skorzystanie z odwrotnej częstości dokumentowej (ang. *inversed document frequency*) opisanej poprzez wzór:

$$idf_t = \log \frac{N}{df_t} \tag{2}$$

gdzie:

N – liczba wszystkich badanych dokumentów,

df_t – częstość dokumentowa – określa liczbę dokumentów dla danego wystąpienia terminu.

Ostatecznie, w celu doprecyzowania schematu ważenia terminów, stosuje się następujący wzór, którego wynik jest iloczynem kartezjańskim odwrotnej częstości dokumentowej i częstości wystąpienia terminu [1][1]:

$$tf - idf_{t,d} = \log \frac{N}{df_t} \cdot tf_{t,d} \tag{3}$$

Wzór (3) nie umożliwia oszacowania podobieństwa, w których liczba dokumentów dla danego wystąpienia terminu jest równa liczbie badanych dokumentów, gdyż logarytm w takim przypadku jest równy 0, co z kolei ma przełożenie na wzory metod bazujących na obliczonych wagach – niedozwolone operacje dzielenia przez 0. Taka sytuacja ma miejsce, gdy np. dokumenty są identyczne. Propozycja modyfikacji wzoru (3) jest następująca:

$$tf - idf_{t,d} = \max(\log \frac{N}{df_t}, 0.00...01) \cdot tf_{t,d} \tag{4}$$

Wprowadzona funkcja zwracająca wartość maksymalną z dwóch argumentów zapobiega otrzymaniu wartości zerowej. Ułamek w drugim argumencie jest dowolnie mały.

3. Metody wyznaczania podobieństwa dla modelów wektorowych

3.1. Współczynniki: Dice i Jaccarda

Oszacowanie wartości podobieństwa dla określonego modelu wektorowego jest odrębnym zagadnieniem. Często do wyznaczenia bliskości pomiędzy wektorami dokumentów danego modelu, stosuję się współczynnik Dice:

$$\text{Dice\_dist}(d_i, d_k) = 2 \frac{\sum_{j=1}^{n} d_{ij} d_{kj}}{\sum_{j=1}^{n} d_{ij}^2 \sum_{j=1}^{n} d_{kj}^2} \tag{5}$$

[1] http://nlp.stanford.edu/IR-book/pdf/06vect.pdf, 119.

lub uogólniony współczynnik Jaccarda:

$$\text{Jaccard\_dist}(d_i, d_k) = \frac{\sum_{j=1}^{n} d_{ij} d_{kj}}{\sum_{j=1}^{n} d_{ij}^2 + \sum_{j=1}^{n} d_{kj}^2 - \sum_{j=1}^{n} d_{ij} d_{kj}} \tag{6}$$

Oba współczynniki umożliwiają obliczenie zgodności pomiędzy cechami wektora.

3.2. Miara cosinusa kąta w funkcji podobieństwa modelu wektorowego

Najbardziej powszechną metodą w dziedzinie analizy podobieństwa dokumentów jest odległość cosinusowa [1][2] (7).

$$\text{Cosine\_dist}(d_i, d_k) = \frac{\sum_{j=1}^{n} d_{ij} d_{kj}}{\sqrt{\sum_{j=1}^{n} d_{ij}^2} \sqrt{\sum_{j=1}^{n} d_{kj}^2}} \tag{7}$$

4. Implementacja algorytmu odległości Levenshteina w procesie ważenia terminów

4.1. Odległość Levenshteina

Odległość Levenshteina umożliwia obliczenie najmniejszej liczby zmian potrzebnych do przeprowadzenia na znakach dwóch porównywanych ciągów, w celu osiągnięcia ich identyczności [4]. Algorytm jest uogólnieniem odległości Hamminga (ang. *Hamming distance*). W przeciwieństwie do odległości Hamminga, odległość Levenshteina umożliwia porównywanie ciągów o różnej długości, co bezpośrednio przekłada się na zastosowanie algorytmu, tj. m.in. w: korekcie pisowni, maszynowym tłumaczeniu tekstów, eksploracji danych tekstowych, a szerzej w mechanizmach wyszukiwarek internetowych, procesorach tekstów itd.

Zasada działania algorytmu polega na porównywaniu znaków mieszczących się na odpowiednich pozycjach w badanych dwóch ciągach, a następnie na tej podstawie uzupełnianiu macierzy o wymiarach adekwatnych do badanych łańcuchów, według poniższego schematu:

$$K = \prod_{x=1}^{N} \prod_{y=1}^{M} D(x,y) = \min(D(x-1,y)+1, D(x,y-1)+1, D(x-1,y-1)+\beta)$$

$$\begin{cases} \beta = 0 : a(x) \equiv b(y) \\ \beta = 1 : a(x) \neq b(y) \\ D(x,0) = x \\ D(0,y) = y \\ D(0,0) = 0 \end{cases} \tag{8}$$

[2] http://nlp.stanford.edu/IR-book/pdf/06vect.pdf, 122.

gdzie:

K – odległość Levenshteina,
$\sum_{x=1}^{N}$ – symbol oznaczający iterację dla $x = (1, ..., N)$,
D – macierz o rozmiarach $N + 1$, $M + 1$, utworzona z dwóch porównywanych ciągów,
N, M – długości badanych ciągów,
$D(x, y)$ – (x, y)-ty element macierzy **D**,
min – funkcja zwracająca wartość najmniejszą z podanych,
β – zmienna przybierająca odpowiednio wartości 0 lub 1,
$a(x)$ – x-ty element ciągu znaków a,
$b(y)$ – y-ty element ciągu znaków b.

4.2. Implementacja funkcji miary podobieństwa ciągów w metodzie ważenia terminów

Algorytm obliczający wagę terminu zawiera instrukcje porównania terminów. Standardowy mechanizm opisany w publikacjach nie uwzględnia możliwości wystąpienia błędów w porównywanych terminach. Istnieją rozwiązania umożliwiające analizę podobnych wyrazów w oparciu o słowniki wyrazów bliskoznacznych (np. projekt WordNet) [2[3], 3], jednak słowniki tego typu nie uwzględniają błędów ortograficznych, jak również wszystkich odmian wyrazów, w szczególności w językach innych niż angielski. Konsekwencją może być niedokładna analiza danych tekstowych.

W celu uwzględnienia możliwych błędów i rzadkich odmian wprowadzona została do algorytmu obliczającego wagę terminu, miara podobieństwa ciągów, oparta na odległości Levenshteina. Miara podobieństwa dwóch ciągów P obliczana jest wg zależności (9):

$$P = 1 - \left(\frac{K}{\max\{N, M\}} \right), \quad \begin{matrix} K \geq 0,\ M > 0,\ N > 0 \\ P \in \langle 0,1 \rangle \end{matrix} \tag{9}$$

gdzie:

$\max\{N, M\}$ – wartość będąca maksymalną liczbą kroków zmieniających jeden tekst w drugi (przypadek, w którym ciągi są całkowicie różne).

Formuła (9) umożliwia otrzymanie miary podobieństwa pomiędzy dwoma terminami. Wynik $P = 1$ oznacza, że dwa porównywane ciągi są identyczne, wynik $P = 0$ oznacza, że ciągi są całkowicie różne. Wartości pośrednie określają stopień podobieństwa.

Poniżej znajdują się pseudokody przedstawiające metodę ważenia terminów z zaimplementowaną miarą podobieństwa ciągów:

```
1 for (c=0; c < Terms.count; c++) {
2 df=0;
3
4     for (j=0; j < d.count; j++) {
5     if (f_exists(d[j].terms, Terms[c])) df++;
6     }
7
8     for (j=0; j < d.count; j++)
9     X[j][c] = f_tf_idf(N, df, f_tf(d[j].terms, Terms[c]));
10}
```

[3] http://webdocs.cs.ualberta.ca/~lindek/papers/acl98.pdf.

gdzie:

- *N* – liczba badanych dokumentów,
- *df* – liczba dokumentów, w których występuje dany termin (ang. *document frequency*),
- *Terms* – tablica unikatowych terminów wchodzących w skład porównywanych wektorów dokumentów,
- *d* – tablica dokumentów, o strukturze przechowującej informacje o numerach dokumentów identyfikowanych przez zmienną *j* oraz występujących w nich terminach,
- *X* – tablica dwuwymiarowa reprezentująca macierz relacji terminów i dokumentów,
- f_exists – funkcja sprawdzająca, czy dany termin występuje w tablicy *d*. Zwraca wartości boolowskie,
- f_tf – funkcja zwracająca liczbę terminów występujących w danym dokumencie (ang. *term frequency*),
- f_tf_idf – funkcja zwracająca wagę terminu, tj. wartość zgodnie ze wzorem (5).

Fragment pseudokodu programu dla funkcji *f_tf*:

```
1 for (i=0; i < d[j].count; i++){
2 if (f_sim_meas(d[j].terms[i], Terms[c]) >= P) ccnt++;}
3    // zamiast d[j].terms[i] == Terms[c]
```

gdzie:

f_sim_meas – funkcja zwracająca wartości zgodne ze wzorem (9).

W powyższym kodzie standardowe porównanie terminów poprzez znaki równości zostało zastąpione funkcją zwracającą miarę podobieństwa pomiędzy terminami. Zmienna *P* w kodzie jest traktowana jako jeden z parametrów analizy danych. Adekwatna sytuacja ma miejsce w kodzie funkcji *exists*.

Fragment pseudokodu programu dla funkcji *f_exists*:

```
1 for (i=0; i < d[j].count; i++){
2    if (f_sim_meas(d[j].terms[i], Terms[c]) >= P)
3    return true;
4 }
5 return false;
```

Fragment pseudokodu funkcji *f_tf_idf*:

```
1 return Math.Max(Math.Log(N / df), 0.0000001) * tf;
```

Wprowadzenie w kodzie funkcji zwracającej wartość maksymalną umożliwia uniknięcie niedozwolonej operacji dzielenia przez zero w funkcji obliczającej odległość cosinusa (7).

4.3. Implementacja mechanizmu obliczeń równoległych w funkcji odległości cosinusa

Użycie algorytmu miary podobieństwa ciągów opartego na odległości Levenshteina pociąga za sobą podwyższenie złożoności obliczeniowej i czasowej. Sposobem na uniknięcie utraty wydajności całego mechanizmu jest zastosowanie technologii obliczeń równoległych. Ze względu na niezależność zdarzeń porównania dokumentów istnieje możliwość przyspieszenia analizy danych poprzez rozłożenie obliczeń na wiele rdzeni procesora(ów).

Poniżej znajduje się pseudokod operacji zrównoleglenia komparacji dokumentów funkcją *cosine_dist*. Algorytm odpowiednio dzieli zdarzenia, jakimi są porównania dwóch dokumentów adekwatnie do liczby rdzeni.

```
1 Parallel.For(0, num_of_proc, sub(nproc) {
2      int int_from, int_to;
3      int_from = nproc * Int(N / num_of_proc) + 1;
4      int_to = (int_from - 1) + Int(N / num_of_proc);
5
6      if (int_to + Int(N / num_of_proc) > N)
7       {int_to = N;}
8
9      for (j_x = int_from ; j_x <= int_to ; j_x++){
10     for (j_y = j_x + 1 ; j_y <= N ; j_y++){
11
12     results[j_y][j_x] = cosine_dist(d, j_y, j_x, Terms.Count);}}
13     End Sub} );
```

gdzie:

- Parallel.For – pętla zrównoleglająca (oparta o rozwiązanie Microsoft.NET [5]), pierwszy parametr jest wartością początkową wskaźnika pętli zrównoleglającej, parametr drugi jest liczbą iteracji kierowanych na poszczególne rdzenie, parametr trzeci jest zestawem procedur,
- *num_of_proc* – liczba rdzeni (procesorów),
- *N* – liczba badanych dokumentów,
- *results* – tablica wyników,
- *nproc* – numer iteracji,
- Int – funkcja zwracająca wartość całkowitą z liczby stałoprzecinkowej,
- *j_x*, *j_y* – zmienne zawierające numery dokumentów do porównania,
- *cosine_dist* – funkcja obliczająca odległość cosinusa według wzoru (7).

Liczbę wszystkich porównań dokumentów, wymaganą dla każdego modelu metody grupowania, można obliczyć ze wzoru na liczbę kombinacji bez powtórzeń:

$$C_N^2 = \binom{N}{2} = \frac{N!}{2!(N-2)!} \tag{10}$$

gdzie:

- 2 – liczba zestawionych dokumentów w jednym porównaniu,
- N – liczba porównywanych dokumentów.

Jak łatwo obliczyć, już dla zaledwie 150 dokumentów liczba porównań wynosi 11,175. Jeżeli przyjąć, że porównania wykonano iteracyjnie bez zrównoleglenia i że jedno porównanie teoretycznie trwa 1 sekundę, to ogólny czas zestawienia wynosi: 3,10 godz.

$$\sum_{nproc=0}^{num\_of\_proc-1} \sum_{c=\text{int}\_from}^{\text{int}\_to} N - c \tag{11}$$

Wzór (11) umożliwia obliczenie liczby porównań dla danego rdzenia (*nproc*) przy znanej liczbie rdzeni (*num_of_proc*) i liczbie dokumentów (*N*). Jeżeli obliczenia byłyby wykonane metodą przedstawioną w pseudokodzie, to w tym przypadku całkowity czas analizy byłby równy najdłuższemu czasowi porównania, czyli 4847 s. (tabela 1). Co daje przyspieszenie o ok. 1,75 godz., czyli obliczenia wykonane zostałyby prawie dwukrotnie szybciej.

Tabela 1

Przykład porównania 150 dokumentów na 4-rdzeniowym procesorze

Rdzeń:	Zakres dokumentów	Liczba porównań dla zakresu	Liczba porównań ogółem
0	1–37	4847	4847
1	38–74	3478	8325
2	75–111	2109	10434
3	112–150	741	11175

5. Test prezentowanych metod i analiza wyników działania algorytmów

W rozdziale przedstawione zostały wyniki testów analizy danych tekstowych z użyciem zaprezentowanych metod na bazie opracowanych symulacji obliczeń.

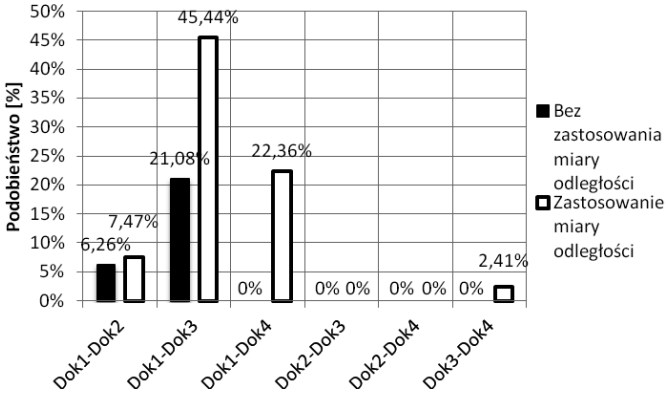
5.1. Implementacja algorytmu miary podobieństwa ciągów

Implementacja algorytmu miary podobieństwa ciągów umożliwia dokładniejsze porównanie dwóch dokumentów tekstowych względem siebie. W celach analizy porównane zostały cztery krótkie zdania (tabela 2) skonstruowane tak aby zawierały wyrazy podobne.

Tabela 2

Zawartość dokumentów poddanych analizie

Nazwa dokumentu	Treść dokumentu
Dokument 1	Odległość Levenshteina w analizie danych tekstowych
Dokument 2	Odległość Hamminga i jej zastosowanie
Dokument 3	Analiza danych tekstowych
Dokument 4	Prezentowany algorytm (autor: V. Levenshtein) analizuje dane tekstowe



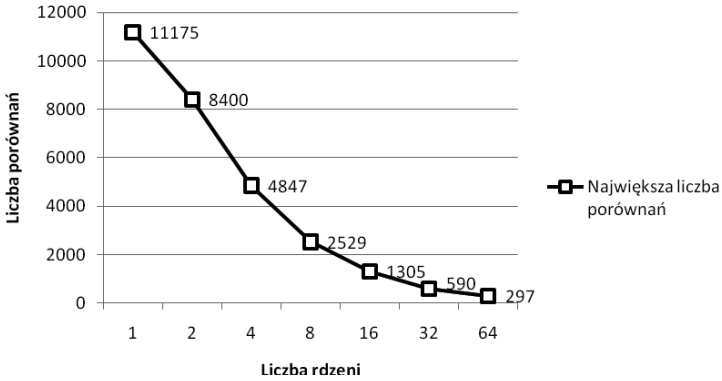
Rys. 1. Wykres przedstawiający wyniki analizy podobieństwa dokumentów (tabela 2)

Fig. 1. Chart describes results of similarities between documents (table 2)

Wykres przedstawia wyniki porównania ciągów bez i z użyciem algorytmu miary podobieństwa ciągów. We wszystkich porównaniach widać znaczy wzrost podobieństwa z użyciem omawianego algorytmu. W ramach testów parametr klasyfikujący ciągi jako podobne wynosił $P >= 0{,}75$ (9).

5.2. Implementacja algorytmu zrównoleglenia operacji porównania dokumentów

Wyniki symulacji analizy dokumentów na wielordzeniowych komputerach przedstawiają wykresy poniżej.

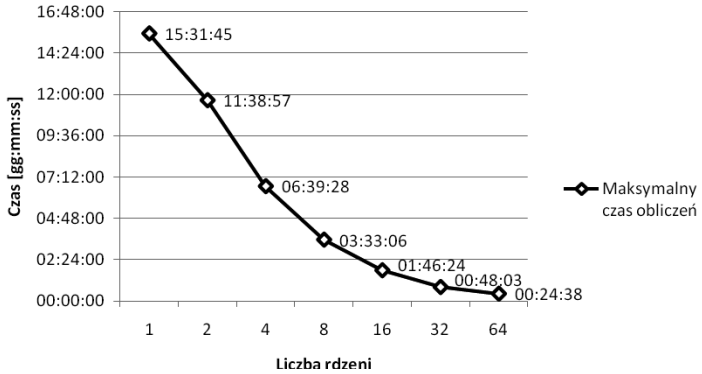


Rys. 2. Wykres zależności liczby rdzeni do maksymalnej liczby porównań dla 150 dokumentów

Fig. 2. Relationship between number of cores and maximum number of comparisons for 150 documents

Wykres na rysunku 2 przedstawia tendencję spadkową największej liczby porównań (tj. dla pierwszego rdzenia) do liczby rdzeni biorących udział w obliczeniach, według modelu zaprezentowanego w rozdziale 4.3.

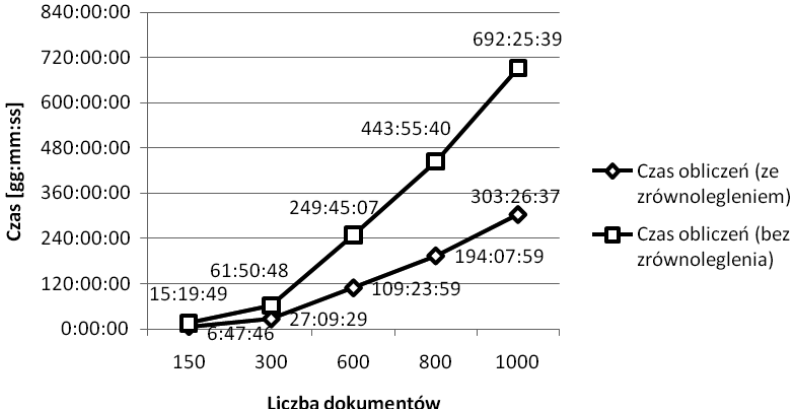
Poniżej przedstawiono wykresy obrazujące czas analizy dużych dokumentów tekstowych o podobnych rozmiarach. Czas porównania dwóch dokumentów w symulacji jest wartością losową z przedziału od 0,5 do 10 sekund.



Rys. 3. Wykres zależności liczby rdzeni do całkowitego czasu obliczeń

Fig. 3. Relationship between number of cores and total computation time

Wykres na rysunku 3 przedstawia tendencję spadkową czasu obliczeń (składających się z komparacji dwóch dokumentów) zbioru 150 dokumentów do liczby rdzeni.



Rys. 4. Czas porównania różnej liczby dokumentów na 4-rdzeniowym procesorze

Fig. 4. Computation time of comparison the different number of documents for 4-core processor

Wykres na rysunku 4 przedstawia różnicę w czasach analizy zmiennej liczby dokumentów na komputerze 4-rdzeniowym.

Zaprezentowane symulacje ukazują istotny wpływ użycia technologii obliczeń równoległych w analizie danych tekstowych na szybkość obliczeń. Zastosowanie opisanej metody podziału znacząco przyspiesza całość procesu obliczeniowego.

6. Podsumowanie

Przedstawiony w publikacji mechanizm analizy danych tekstowych oparty na mierze podobieństwa ciągów umożliwia komparację terminów opisanymi metodami w sposób bardziej precyzyjny, z uwzględnieniem odmian i błędów ortograficznych.

Implementacja algorytmu odległości Levenshteina pociąga za sobą podwyższenie złożoności obliczeniowej i czasowej. Rekompensatą za obniżenie wydajności mechanizmu jest przedstawiony sposób na zrównoleglenie analizy dokumentów. Użycie technologii obliczeń równoległych wyraźnie przyspiesza proces oszacowania relacyjności pomiędzy poszczególnymi dokumentami, co dla dużego zbioru danych jest bardzo istotne.

Zastosowanie zarówno algorytmu miary podobieństwa ciągów bazującego na odległości Levenshteina, jak również zrównoleglenie obliczeń znacząco optymalizuje całość procesu obliczeniowego.

Literatura

[1] Manning C.D., Raghavan P., Schutze H., *Introduction to Information Retrieval*, Cambridge University Press, 2007.

[2] Lin D., *Automatic retrieval and clustering of similar words*, COLING 1998, ACL, 1998, 768-774.

[3] Piasecki M., Broda B., *Semantic similarity measure of Polish nouns based on linguistic features*, Business Information Systems 10th International Conference, Poznań, *Lecture Notes in Computer Science*, vol. 4439, Springer, 2007.

[4] Левенштейн В.И., *Двоичные коды с исправлением выпадений, вставок и замещений символов*. Доклады Академий Наук СССР 163 (4), 845-848.

[5] Campbell C., Johnson R., Miller A., Toub S., *Parallel Programming with Microsoft .NET. Design Patterns for Decomposition and Coordination on Multicore Architectures*, Microsoft Press, 2010.