Wykrywanie Istotnych i Nieistotnych Fragmentów Stron WWW

Michał Wójcik

Instytut Informatyki, Wydział Elektroniki i Technik Informacyjnych, Politechnika Warszawska, ul. Nowowiejska 15/17,
00-665 Warszawa, Poland
michalwojcik@onet.pl

Streszczenie. Na większości stron WWW możemy wyróżnić fragmenty istotne i nieistotne. Do tych pierwszych zaliczamy wszystkie te części strony, dla których jest ona przeglądana. Wśród tych drugich znajdziemy wszystkie te fragmenty, których brak w danym przypadku nie wpłynie na merytoryczną wartość strony. W poniższym artykule przedstawiono algorytmy WISDOM, MDR i IKM, jako wybrane metody odnajdowania fragmentów istotnych. Ponadto, opisano sposób automatycznego znajdowania i usuwania szablonów ze stron WWW. Omówiono również zarys autorskiego algorytmu pozwalającego wykrywać na stronach WWW różne klasy obiektów. I wreszcie, przedstawiono projekt narzędzia stosującego w praktyce powyższe rozwiązanie.

Słowa kluczowe: eksploracja danych, wyszukiwanie informacji, fragment istotny, fragment nieistotny

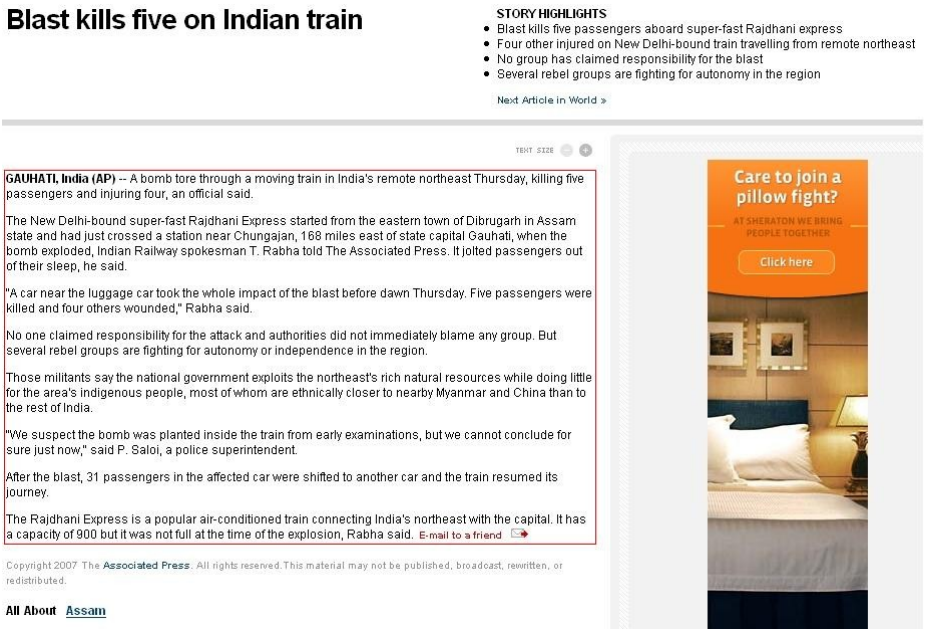
1 Wprowadzenie

Biorąc pod uwagę ogromną liczbę stron WWW dostępnych w Internecie, coraz większe znaczenie odgrywa automatyczne znajdowanie na tych stronach fragmentów istotnych i nieistotnych. Omawianie zagadnienia będącego tytułem poniższej pracy warto rozpocząć od odpowiedzi na pytanie, czym tak naprawdę są te „istotne" i „nieistotne" fragmenty oraz w jakim celu należy je wykrywać.

1.1 Opis problemu

Każda osoba lub pająk webowy otwierający stronę WWW robi to w celu odczytania tam jakichś konkretnych treści. W przypadku np. internetowego serwisu informacyjnego może to być artykuł (por. rys. 1.1.1), zaś na osobistej stronie WWW - opublikowane zdjęcia z imprezy rodzinnej. Kolejnym przykładem może być lista odnośników znajdująca się na stronie wynikowej wyszukiwarki internetowej (por. rys. 1.1.2), a także lista opisów produktów umieszczona w witrynie sklepu

internetowego. Dalej fragmenty (bloki, elementy) te będą nazywane istotnymi (pożądanymi, relewantnymi).



Rys. 1.1.1. Przykładowa strona zawierająca artykuł prasowy.

Poza tymi fragmentami, dla których strona WWW jest przeglądana, można również wyróżnić na niej te, które nic nie wnoszą pod względem merytorycznym i równie dobrze mogłoby ich nie być wcale. Przykładem takich treści są paski nawigacyjne, loga, liczniki, nagłówki stron, stopki zawierające informacje o prawach autorskich itd. Są one fragmentami nieistotnymi. Oczywiście może się zdarzyć, że ktoś przegląda strony internetowe w poszukiwaniu, dajmy na to, banerów reklamowych. Celem takiego zachowania może być chęć utworzenia własnej reklamy po uprzednim zapoznaniu się z przykładowymi banerami zamieszczonymi w sieci. W takim przypadku fragmentami istotnymi na stronach są właśnie reklamy, a wszystko inne stanowi fragmenty nieistotne.

Zdarza się, że nie jest z góry powiedziane, jakie treści stron są pożądane. Rozwiązanie tego problemu sprowadza się do realizacji dwóch zadań. W pierwszej kolejności konieczne jest samodzielne ustalenie, co na danej stronie jest jej treścią, a co stanowi „szum informacyjny". Należy pamiętać o tym, że określenie tego, co na danej stronie może być istotne nie musi oznaczać wskazania konkretnych bloków. Celem kroku pierwszego jest ogólne zbadanie strony i dobranie algorytmów przeznaczonych do wyszukiwania konkretnych treści, zaś kolejnym, drugim krokiem jest skuteczne wyłuskanie fragmentów relewantnych za pomocą metod wskazanych w kroku poprzednim. Jest to bez wątpienia najbardziej interesujący i jednocześnie skomplikowany wariant problemu wykrywania fragmentów istotnych i dlatego właśnie jemu zostaną poświęcone dalsze rozdziały poniższego artykułu.

Rys. 1.1.2. Przykładowa strona zawierająca rezultat pracy wyszukiwarki internetowej.

Podsumowując, blokiem istotnym może być właściwie każda część strony WWW. To, który fragment będzie uznany za istotny, zależy od określonego przypadku omawianego problemu.

1.2 Motywacja do pracy

Powodów, dla których warto tworzyć systemy automatycznie wykrywające istotne lub nieistotne fragmenty stron WWW, jest bardzo wiele, a poniżej zostaną omówione te najważniejsze.

Rozważmy serwis WWW, na którym strony są generowane w sposób dynamiczny, w oparciu o skrypty. Jeżeli weźmiemy pod uwagę pojedynczą stronę występującą w tym serwisie, to z łatwością stwierdzimy, że nie cała strona jest rezultatem działania skryptów. Pewne elementy są statyczne, nie wnoszą nic pod względem merytorycznym i powtarzają się na wszystkich lub na większości stron należących do tego serwisu. Te i inne strony często są przetwarzane przez pająki webowe należące do systemów cache'ujących. Jeżeli jakiś element powtarza się na wielu stronach to warto, aby owe systemy zapamiętywały go tylko raz, a nie w liczbie równej ilości stron, na których się pojawił. Oczywiście, aby było to możliwe, potrzebne jest narzędzie, które taki element, w tym przypadku nieistotny (bo niemerytoryczny), zidentyfikuje. Jednokrotne zapamiętywanie powtarzających się elementów pozwala na oszczędzanie przestrzeni dyskowej.

Innym przykładem systemu, w którym narzędzie do wykrywania elementów istotnych i nieistotnych może znaleźć zastosowanie, są wyszukiwarki internetowe, w tym np. serwisy zwracające łącza do artykułów prasowych, takie jak np. Google News, czy Altavista News. Jeżeli na świecie dojdzie do jakiegoś kataklizmu, to z całą pewnością wszystkie agencje prasowe opracują notki opisujące to zdarzenie. Artykuły te pojawią się na łamach wszystkich internetowych serwisów informacyjnych. Jeżeli

teraz w wyszukiwarce artykułów prasowych zostaną wpisane słowa kluczowe związane z owym kataklizmem, to w rezultacie system zwróci listę łącz do różnych serwisów informacyjnych. W wielu przypadkach artykuły zamieszczone na łamach tych serwisów będą się powtarzać, bo będą pochodziły z tej samej agencji prasowej. Gdyby jednak przed rozpoczęciem indeksowania stron zawierających artykuły odrzucono wszystkie inne elementy poza samym artykułem, to wówczas taka wyszukiwarka byłaby w stanie wykrywać duplikaty. Wówczas dla każdej pojedynczej notki z agencji prasowej mogłaby indeksować tylko jeden serwis ją zawierający. Oznaczałoby to, że każde ze zwróconych hiperłącz prowadziłoby do innego artykułu. Zatem użycie systemu klasyfikującego bloki na stronach WWW pozwoliłoby również na ograniczenie rozmiaru indeksów wyszukiwarek, a także umożliwiłoby zwiększenie szybkości ich tworzenia.

Kolejnymi odbiorcami rozważanego narzędzia są producenci oprogramowania stosowanego w urządzeniach wyposażonych w małe wyświetlacze, a więc najczęściej przenośnych. Niewielkie ekrany często oferują niskie rozdzielczości. Zatem podczas przeglądania sieci WWW za pomocą tych urządzeń znaczącym udogodnieniem byłoby automatyczne znajdowanie i pokazywanie na wyświetlaczach tych urządzeń jedynie najistotniejszych treści przeglądanych stron.

Jak widać jest wiele powodów, dla których warto automatycznie odróżniać na stronach WWW fragmenty istotne od nieistotnych. W dalszej części pracy zostaną przedstawione przykładowe algorytmy realizujące to zadanie, a także szkic rozwiązania autorskiego.

2 Podobne prace

Istnieje wiele metod, w oparciu o które podejmowane są próby opracowania narzędzi znajdujących elementy relewantne. Najprostszym, „amatorskim" rozwiązaniem jest aplikacja, która dokonuje poszukiwań w oparciu o zaszyte w jej kodzie wzorce, będące rezultatem obserwacji kodu źródłowego strony WWW przez programistę. Taka metodyka jest nieefektywna z dwóch powodów. Po pierwsze, za każdym razem, gdy na stronie zmieni się sposób prezentacji treści, konieczna jest modyfikacja kodu aplikacji. Jak wiadomo, współcześni webmasterzy często modyfikują wygląd strony w celu jej uatrakcyjnienia. Po drugie, gdy rodzi się konieczność prowadzenia odkryć na kolejnych stronach o innym sposobie prezentacji treści, wtedy również konieczne jest przebudowanie aplikacji przez programistów.

Innym podejściem jest zastosowanie wyuczonego klasyfikatora. W tym przypadku konieczny jest podział stron pochodzących z jednej witryny na dwa zbiory – treningowy i „testowy". Na stronach należących do pierwszego zbioru niezbędne jest manualne wyróżnienie poszczególnych fragmentów i określenie czy są istotne, czy też nie. Narzędzie z zaimplementowanym klasyfikatorem, takim jak np. Naiwny Klasyfikator Bayesa lub Support Vector Machine dzieli oznaczone strony na bloki i dokonuje ekstrakcji ich cech. Na podstawie tego i informacji o tym, które bloki są istotne, narzędzie uczy się jak klasyfikować pozostałe strony tzn. te, które należą do zbioru „testowego". Ten typ narzędzi został szerzej scharakteryzowany w [1]. Konkretny przykład systemu uczącego się, opartego na klasyfikatorze SVM można

odnaleźć w [2]. Niestety, w tym przypadku jak i w przedstawionym w poprzednim akapicie, zmiana budowy strony pochodzącej z „wyuczonej" witryny jak i wybór nowej witryny do rozpoznawana pociąga za sobą konieczność ponownego podziału stron i oznaczenia bloków na tych, które należą do zbioru testowego.

Zdecydowanie najlepszą koncepcją rozwiązania omawianego problemu jest metoda w pełni automatyczna, a więc taka, która pozwala na przeprowadzenie procesu wyszukiwania bez udziału czynnika ludzkiego, jedynie w oparciu o treść zadanej strony WWW lub ich zbioru. Do tej pory powstało wiele tego typu rozwiązań, jednak na szczególną uwagę zasługują trzy, względnie nowe metody przedstawione poniżej.

2.1 WISDOM

Autorzy [3] zasugerowali metodę rozpoznawania bloków zawierających dużą ilość tekstu (np. artykułów prasowych), a także posiadających budowę Table of Contents (TOC), czyli posiadających prostą strukturę rekordową. Swoje rozwiązanie nazwali Web Intra-page Informative Structure Mining based on Document Object Model, w skrócie WISDOM. Pojedynczy przebieg algorytmu wiąże się z przetwarzaniem wielu stron należących do tej samej witryny, przy czym sam proces wykrywania jest przeprowadzany oddzielnie dla każdej ze stron. Wynikiem działania algorytmu jest drzewo znaczników HTML opisujące strukturę fragmentów istotnych, zwane Informative Structure (IS).

Algorytm, począwszy od strony o adresie zadanym na wejściu, wczytuje strony do pewnej, z góry zadanej głębokości. Następnie, dla każdej z nich generuje strukturę DOM, szerzej opisaną w [4]. Otrzymana w ten sposób struktura nosi nazwę Information Coverage Tree (ICT). Jednocześnie, dla każdego z odnalezionych w tekście swobodnym słów obliczana jest w sposób iteracyjny entropia, a na jej podstawie ważkość słowa. W oparciu o wstępujący algorytm, każdemu węzłowi ICT przypisywane są współczynniki: ALEN, CLEN oraz API. Ten pierwszy oznacza wyrażoną w znakach długość tekstu swobodnego przypisanego do węzła ICT i ograniczonego znacznikami *<a>...*. Drugi to długość pozostałego tekstu swobodnego. Ostatni współczynnik to uśredniona ważkość tych słów, które występują w tekście ograniczonym znacznikami *<a>...* i jednocześnie na stronach wskazywanych przez te znaczniki. Na podstawie wartości tych współczynników określana jest i przypisywana każdemu węzłowi innemu niż liść wartość parametru SII. Jest ona tym wyższa, im bardziej wartości trzech wcześniej wspomnianych parametrów dla bezpośrednich potomków danego węzła są zbliżone do siebie. Im większe SII, tym lepiej, gdyż oznacza to, że poddrzewo ICT wyznaczone przez dany węzeł posiada regularną strukturę.

Kolejnym etapem jest zastosowanie zachłannego algorytmu zstępującego, mającego na celu odnalezienie co najwyżej k węzłów ICT, które odznaczają się największą wartością SII i jednocześnie takich, że wartość SII przekracza określony próg i spełnione są pewno dodatkowe heurystyki. Spełnienie tych ostatnich wskazuje na klasę obiektu, który opisany jest poddrzewem o korzeniu w rozważanym węźle. Nietrudno zauważyć, że wartość tego progu ma kluczowe znaczenie dla liczby i

jakości zwróconych k bloków. Rezultatem tej fazy algorytmu jest szkielet IS, a więc minimalne drzewo łączące w całość wszystkie poddrzewa znalezionych bloków.

Ostatnim etapem WISDOM jest zastosowanie dwóch operacji łączenia poddrzew, których celem jest uporządkowanie całej struktury i wygenerowanie ostatecznej wersji IS.

2.2 Intelligent Knowledge Mining

W [5] zaproponowano metodę rozpoznawania fragmentów stron WWW o strukturze rekordowej, czyli tzw. stron systematycznych. Autorzy nazwali ją Intelligent Knowledge Mining. Jest to rozszerzenie innego, wcześniej zaproponowanego rozwiązania, zwanego Mining Data Records [6].

MDR pozwalał na wskazanie na stronie WWW tych bloków, które zawierają rekordy, a następnie umożliwiał ich wyłuskanie. Proces znajdowania był realizowany w kilku krokach. Pierwszy etap owocował utworzeniem drzewa znaczników HTML, z wyłączeniem tych tagów które, zdaniem autorów, jedynie utrudniają wyszukiwanie np. <*script*> lub <*!*--. Kolejny krok wiązał się z odnalezieniem w rozważanym drzewie węzłów uogólnionych (generalized nodes) i obszarów informacyjnych (data regions). Węzeł uogólniony to zbiór węzłów drzewa znaczników mających wspólnego przodka i sąsiadujących ze sobą. Obszar informacyjny jest to zbiór dwóch lub więcej węzłów uogólnionych mających wspólnego przodka, zawierających tyle samo węzłów, sąsiadujących ze sobą i takich, że znormalizowana odległość edycyjna pomiędzy każdą parą sąsiadujących ze sobą węzłów uogólnionych jest dostatecznie mała. Do obliczeń odległości brane były łańcuchy znaczników wchodzących w skład poddrzew wyznaczanych przez dany węzeł uogólniony. Następny etap wiązał się z odnalezieniem rekordów w każdym obszarze informacyjnym, albowiem nie zawsze węzły uogólnione wchodzące w jego skład odpowiadały rekordom. Mogło się zdarzyć, że jeden taki węzeł odpowiadał dwóm rekordom. Jednocześnie algorytm wykrywał pojedyncze rekordy, które nie należą do danego obszaru informacyjnego, ale bezpośrednio z nim sąsiadują i mają podobną strukturę do tych obiektów, które do tego obszaru należą.

IKM pozwala dodatkowo na ustalenie stopnia przydatności tych bloków. Jeżeli na stronie występuje więcej niż jeden blok o strukturze rekordowej, to dzięki temu możliwe jest wskazanie bloków najistotniejszych. Jeżeli obecny jest tylko jeden, to można go przyjąć bądź odrzucić porównawszy wynik działania IKM z predefiniowanym progiem. Algorytm omawiany w poniższym podrozdziale w pojedynczym przebiegu przetwarza tylko jedną stronę. Jego główną ideą jest zastosowanie wobec każdego bloku o strukturze rekordowej trzech metryk: podobieństwa (similarity), gęstości (density) i zróżnicowania (diversity).

Celem pomiaru podobieństwa jest ustalenie, w jakim stopniu węzły uogólnione wchodzące w skład analizowanej grupy są do siebie podobne. W tym celu badana jest znormalizowania odległość Levenshteina pomiędzy każdymi dwoma sąsiadującymi wzorcami. Na jej podstawie obliczany jest w/w współczynnik. Im mniejsze odległości, tym podobieństwo jest większe.

Gęstość to odsetek obszaru zajmowanego przez wszystkie rekordy w bloku, do którego należą. Większy odsetek oznacza wyższą wartość wspomnianego wskaźnika.

Ostatnim współczynnikiem jest zróżnicowanie, czyli stopień zróżnicowania wnętrza poszczególnych wzorców regularnych wchodzących w skład grupy. Obliczenie tego współczynnika polega na zbadaniu zawartych w nich tzw. Items' Styles (IS). Items' Style jest to łańcuch znaków utworzony z nazw znaczników łączących w drzewie DOM danego węzła uogólnionego korzeń z jednym z liści. W celu ustalenia zróżnicowania badane jest to jak wiele IS powtarza się w danym węźle. Najkorzystniejsza sytuacja ma miejsce wtedy, gdy każdy IS jest inny. Wówczas zróżnicowanie osiąga wartość maksymalną.

Poziom istotności jest zdefiniowany jako iloczyn powyższych trzech cech.

2.3 Automatyczne znajdowanie szablonów

Interesujący algorytm przeznaczony dla dużych wyszukiwarek internetowych został przedstawiony w [7]. Pozwala on na znajdowanie powtarzających się fragmentów stron WWW należących do jednej witryny. Podczas jednego przebiegu tego algorytmu przetwarzanych jest wiele stron jednocześnie. Wynikiem działania tego algorytmu jest zbiór wejściowy z usuniętymi elementami wspólnymi. Dla każdej strony ze zbioru wejściowego tworzone jest drzewo znaczników HTML. Następnie, wszystkim blokom przypisywane są sekwencje liczb oznaczające ich pozycję w drzewie. Po uzyskaniu drzew dla wszystkich przetwarzanych stron bloki pochodzące ze wszystkich drzew są poddawane grupowaniu, przy czym do jednej grupy mogą należeć tylko te bloki, które mają tą samą pozycję oraz takie same wartości atrybutów odpowiadających im znaczników HTML. Otrzymane w ten sposób grupy noszą nazwę block-style clusters (BSC).

Kolejnym etapem jest utworzenie odwrotnego indeksu słów występujących na stronach. Każdemu słowu umieszczonemu w indeksie przyporządkowywana jest lista par składających się ze wskazania na blok, w którym dane słowo wystąpiło oraz tzw. binarnej listy wystąpień, określającej pozycję słowa w bloku. Lokalizacja słowa wyrażona jest w postaci 32 bitowej liczby. Po utworzeniu indeksu bloki wchodzące w skład pojedynczej listy są grupowane, przy czym do jednej mogą być zaliczone tylko te, które należą do tego samego BSC i które posiadają dostatecznie podobne listy wystąpień. Podobieństwo dwóch pozycji określa się za pomocą tzw. bloom filtera. Uzyskane w tym kroku bloki noszą nazwę word-feature cluster (WFC).

Ostatnim etapem jest zbadanie, czy słowa umieszczone w indeksie mają dostatecznie liczne WFC w obrębie przypisanych list. Jeżeli tak, to takie słowo jest uznawane za „szablonowe". Teraz, jeżeli blok zawiera dostatecznie duży odsetek takich słów, to jest on usuwany ze stron należących do zbioru wejściowego.

3 Propozycje rozwiązań

Celem pracy magisterskiej omawianej w poniższym artykule jest opracowanie możliwie najbardziej uniwersalnego i elastycznego narzędzia przeznaczonego do wykrywania fragmentów istotnych. Szczególny nacisk kładziony jest na szybkość działania, gdyż jest ono dedykowane do zastosowań automatycznych, tzn. w przyszłości będzie ono częścią większego systemu informatycznego.

Najistotniejszą klasą rozpoznawanych obiektów są tutaj bloki o strukturze rekordowej. Stąd też, ważnym elementem systemu będą rozwiązania wykorzystane w systemach MDR oraz IKM, gdyż zgodnie z wynikami eksperymentów opublikowanych przez ich autorów dają one bardzo zadowalające rezultaty. Implementacja kluczowych fragmentów użytych tam algorytmów będzie okazją, aby stwierdzić, czy tak jest naprawdę.

Równie istotnym elementem narzędzia będzie wykrywanie dużych fragmentów tekstowych, dlatego też istotną rolę odegrają pomysły wykorzystane w systemie WISDOM. Wadą tego rozwiązania jest to, że wykorzystuje on stemmer Portera, który jak wiadomo jest przeznaczony tylko dla języka angielskiego.

I wreszcie, usuwanie fragmentów „zdecydowanie" nieistotnych odegra znaczącą rolę w procesie wykrywania tych istotnych, gdyż pozwali na skrócenie czasu niezbędnego do przeprowadzenia znajdowania tych ostatnich. W narzędziu możliwość usuwania szablonów będzie występowała jako element opcjonalny, ponieważ w niektórych sytuacjach może ona powodować, paradoksalnie, wydłużenie czasu wyszukiwania. Ponadto w przypadku, gdy w grę wchodzi prowadzenie wyszukiwania na pojedynczych stronach WWW, omówiony wyżej algorytm usuwający szablony nie zdaje egzaminu.

Każdy projekt magisterski powinien zawierać elementy twórcze. W tym, który jest omawiany w poniższym rozdziale, skupiono się na następujących dwóch kwestiach, dotąd niepodnoszonych w literaturze. Po pierwsze, wszystkie dotychczas zaproponowane rozwiązania znajdujące bloki z rekordami nie analizują treści samych rekordów. Chodzi o to, że gdy np. mamy pierwszą stronę wynikową wyszukiwarki internetowej, na której znajduje się dziesięć łącz najbardziej pasujących do zadanej kwerendy, to wspomniane algorytmy zwrócą jedynie dziesięć bloków zawierających oczywiście właściwe odnośniki, ale również komentarze i łącza np. „wersja HTML", „podobne strony", etc. (patrz rys. 3.1) Celem omawianego projektu magisterskiego jest wprowadzenie dodatkowych mechanizmów, które pozwolą na wyłuskanie z rekordów właściwych treści, którymi w tym przypadku są łącza wynikowe.

Rys. 3.1. Pojedyncze rekordy będące rezultatem pracy wyszukiwarki internetowej.

Drugim elementem będzie zastosowanie heurystyk mających na celu określenie klasy przetwarzanej strony WWW. Zgodnie z założeniami, narzędzie może otrzymać

na wejściu dowolną stronę, co oznacza, że w celu wykrycia wszystkich rodzajów obiektów pożądanych konieczne jest zastosowanie wszystkich algorytmów, z których każdy przeznaczony jest do znajdowania obiektów innego typu. Stąd też, słusznym krokiem byłoby wprowadzenie etapu wstępnego przetwarzania strony, w ramach którego nastąpiłoby ustalenie tych klas bloków, które z największym prawdopodobieństwem mogą pojawić się na stronie i na tej podstawie zastosowanie jedynie wybranych algorytmów wyszukujących. Pozwoli to ograniczyć całkowity czas przetwarzania strony.

Tak jak w przypadku wielu innych problemów z dziedziny wyszukiwania informacji celem jest, aby otrzymany zbiór bloków istotnych był jak najwyższej jakości. Dobrym miernikiem są tutaj współczynniki szeroko stosowane w dziedzinie IR, czyli precision (PR), recall (R) i F-measure (F) (por. wzory 4.1, 4.2 i 4.3).

$$PR = \frac{|ds \cap r|}{|ds|} \quad \textbf{(4.1)}$$

$$R = \frac{|ds \cap r|}{|dr|} \quad \textbf{(4.2)}$$

$$F = \frac{2 \cdot R \cdot PR}{R + PR} \quad \textbf{(4.3)}$$

dr – obiekty relewantne; ds - obiekty uznane przez narzędzie za relewantne

W idealnym przypadku tzn. takim, w którym program wykrywa wszystkie właściwe obiekty i tylko te właściwe, wszystkie trzy powyższe współczynniki wynoszą 1. Użycie powyższych miar w omawianym projekcie jest słuszne również dlatego, że biblioteka będzie jedynie dzieliła fragmenty stron na istotne lub nie, bez wprowadzania żadnych pośrednich kategorii.

4 Opis aplikacji

Narzędzie wyszukujące jest realizowane w języku C++ i będzie dostarczone jako biblioteka DLL. Przyczyną wyboru takiej formy implementacji jest potrzeba uzyskania programu odznaczającego się jak największą szybkością działania, której nie można by było osiągnąć w środowisku .NET czy Java. Ponadto, C++ zapewnia większą przenośność, gdyż nie wymaga obecności platform takich, jak te wymienione wyżej. Biblioteka łączona dynamicznie pozwoli na łatwe podpięcie jej do większego projektu. Narzędzie będzie pracowało w oparciu o kod źródłowy strony lub stron, przekazany podczas wywołania metod w nim zaimplementowanych. Przetworzone

źródło strony z oznaczonymi fragmentami istotnymi i zaznaczonymi lub w ogóle odrzuconymi blokami nieistotnymi będzie zwracane przez metody biblioteki.

Program demonstracyjny implementowany jest za pomocą języka C# i w oparciu o interfejs Windows Forms. W tym przypadku nie liczy się już prędkość działania ani przenośność, lecz możliwość łatwego i szybkiego opracowania graficznego interfejsu użytkownika, pozwalającego na zaprezentowanie rezultatów działania narzędzia wykrywającego. Istotną cechą tego programu będzie możliwość łatwej zmiany parametrów pracy w/w biblioteki.

W celach testowych użytkownik będzie miał możliwość oznaczenia za pomocą specjalnego tagu tych fragmentów, które wg niego są istotne. Gdy biblioteka zakończy pracę z kodem strony, wtedy program demonstracyjny porówna bloki wyróżnione przez nią i te, które zostały wskazane przez człowieka, a następnie obliczy wspomniane w poprzednim rozdziale współczynniki jakości.

5 Podsumowanie

Jak wspomniano we wstępie, narzędzie automatycznie rozpoznające różne klasy bloków z całą pewnością wzbogaci wiele systemów informatycznych. Będzie tym bardziej wartościowe, im skuteczniejsze, szybsze i bardziej przenośne.

Bibliografia

1. Xiao, X., Luo, Q., Xie, X., Ma, W.Y.: A Comparative Study on Classifying the Functions of Web Page Blocks. CIKM'06, Arlington (2006)
2. Song, R., Liu, H., Wen, J.R., Ma, W.R.: Learning Block Importance Models for Web Pages. WWW2004, New York (2004)
3. Kao, H.Y., Ho, J.M., Chen, M.S.: WISDOM: Web Intra-page Informative Structure Mining based on Document Object Model. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 5 (2005)
4. W3C DOM, http://www.w3.org/DOM/
5. Tseng, Y.F., Kao H.Y.: The Mining and Extraction of Primary Informative Blocks and Data Objects from Systematic Web Pages. IEEE/WIC/ACM International Conference on Web Intelligence (2006)
6. Liu, B., Grossman, R., Zhai, Y.: Mining Data Records in Web Pages. UIC Technical Report (2003)
7. Chen, L., Ye, S., Li, X.: Template Detection for Large Scale Search Engines. SAC'06, Dijon (2006)