„Wykrywanie istotnych i nieistotnych fragmentów stron WWW”

Michał Wójcik

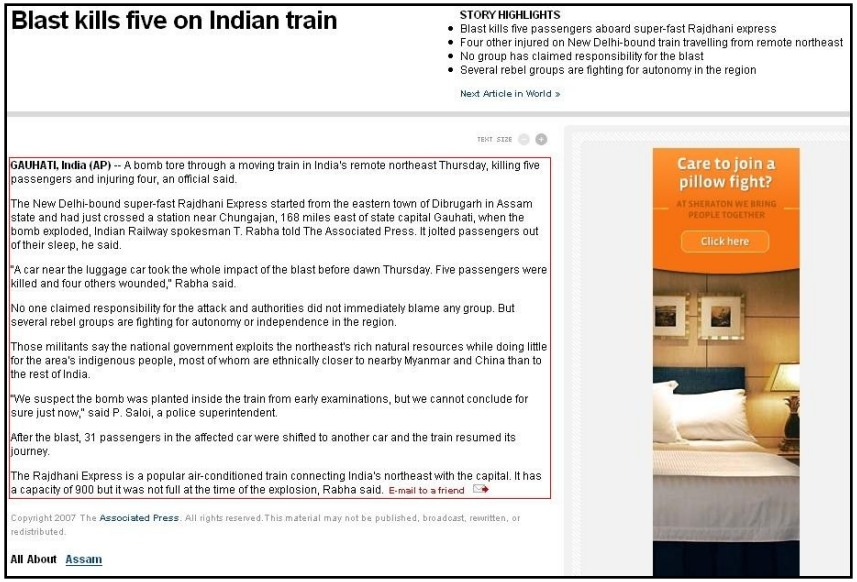
opiekun naukowy: dr inż. Piotr Gawrysiak

Plan prezentacji

- definicja fragmentów istotnych i nieistotnych, oraz powody dla których warto je wykrywać
- rodzaje podejść do realizacji w/w tematu
- przykłady algorytmów wykrywających bloki:
  - z artykułami i TOC
  - o strukturze rekordowej
  - z szablonami
- podsumowanie

Które fragmenty są „istotne”, a które nie są?

- użytkownik przegląda strony WWW w poszukiwaniu określonych treści
- treścią mogą być np. artykuły prasowe, rezultaty działania wyszukiwarek, opisy artykułów w sklepie internetowym
- strony WWW z taką zawartością oferują również liczne dodatki tj. reklamy, paski nawigacyjne, nagłówki, stopki etc.
- pod względem merytorycznym nie wnoszą one nic i nie stanowią wartości dla takiego odbiorcy



W jakim celu wyróżniać fragmenty istotne?

- ograniczanie wymaganej przestrzeni dyskowej potrzebnej do przechowywania stron WWW
- ograniczanie rozmiarów indeksów wyszukiwarek i zwiększanie jakości rezultatów wyszukiwania
- przystosowywanie stron WWW do wyświetlania na urządzeniach przenośnych
- znajdowanie ściśle określonych klas bloków

W jaki sposób wyróżniać fragmenty istotne?

- w oparciu o „sztywne” reguły wydedukowane przez programistę na podstawie obserwacji kodu HTML strony
- poprzez zastosowanie klasyfikatora np. NKB, SVM, wyuczonego na zbiorze treningowym zawierającym oznaczone przez użytkownika bloki
- dzięki wdrożeniu w pełni automatycznych metod nie wymagających ingerencji użytkownika

WISDOM: Web Intra-page Informative Structure Mining based on DOM

- przeznaczony do wykrywania fragmentów stron WWW zawierających artykuły oraz tzw. Table of Contents (TOC)
- podczas pojedynczego przebiegu algorytmu przetwarzane są strony należące do tej samej witryny, przy czym wykrywanie elementów istotnych dla każdej z nich realizowane jest niezależnie
- rezultatem działania jest drzewo (drzewa) znaczników HTML opisujące strukturę fragmentów istotnych (tzw. IS – Informative Structure)

WISDOM – etapy algorytmu

1) wczytanie dostatecznie dużej liczby stron z pojedynczej witryny internetowej, utworzenie dla każdej z badanych stron struktury DOM i obliczenie stosownych parametrów
2) znalezienie k bloków o największej „wartości merytorycznej”
3) korekta struktury znalezionych k bloków

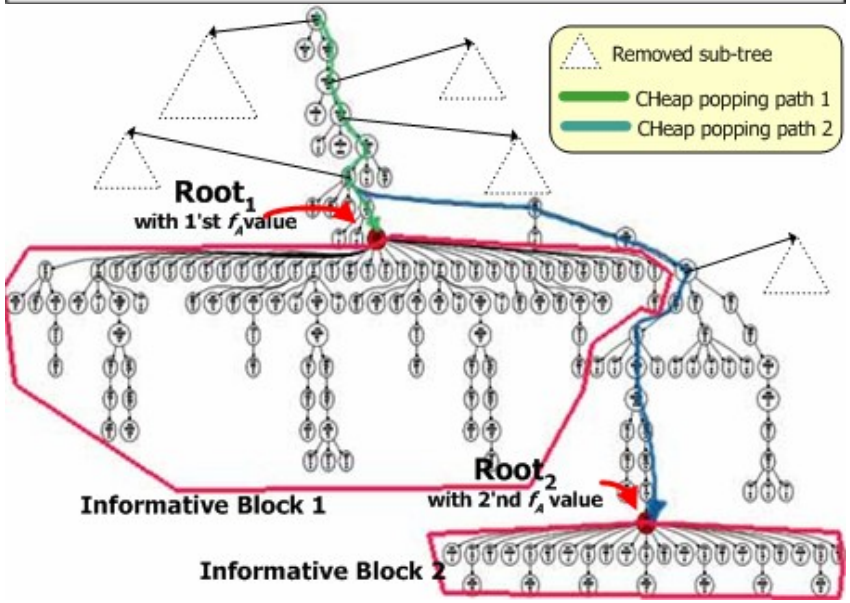
WISDOM – 1. etap

- strony należące do witryny są wczytywane zgodnie z zadaną głębokością
- w oparciu o tekst swobodny pochodzący ze stron ustalana jest ważkość każdego ze słów (ujemna zależność liniowa od entropii)
- tworzona jest struktura DOM dla strony (stron) przeznaczonej do wykrywania
- dla każdego węzła tej struktury liczone są współczynniki:
  - ALEN – długość tekstu przypisanego do węzła i ograniczonego znacznikami <a>…
  - CLEN – długość pozostałego tekstu przypisanego do węzła
  - API $$API(N) = \sum_{j=1}^{m} \frac{1}{EN(term_j)}$$

WISDOM – 2. etap

- na podstawie otrzymanego drzewa budowana jest struktura ICT (Information Coverage Tree) w oparciu o algorytm wstępujący:
  - dla każdego węzła liczona jest zagregowana wartość współczynników ALEN, CLEN oraz API
  - na postawie w/w określana jest wartość parametru SII
- w oparciu o algorytm MIB realizowane jest zstępujące przeszukiwanie ICT w celu ustalenia k najbardziej znaczących bloków (k jest predefiniowane)

```
Algorithm MIB (k, f_A, ST) begin
/* Cheap is a sorted stack */
 1:InfoBlock = 0
 2:Push root node into CHeap(f_A)
 3:While (InfoBlock < k and CHeap is not empty) begin
 4:        Pop Node N with max(f_A) from CHeap(f_A)
 5:        If (SII(N, f_A) > ST or N is a leaf) then
 6:                find = true
 7:                If (N matches the type constrain) then
 8:                        insert N into CandidateSet
 9:                        InfoBlock = InfoBlock + 1
10:               end if
11:       else
12:               push children(N) into CHeap(f_A)
13:       end if
14:end
End
```



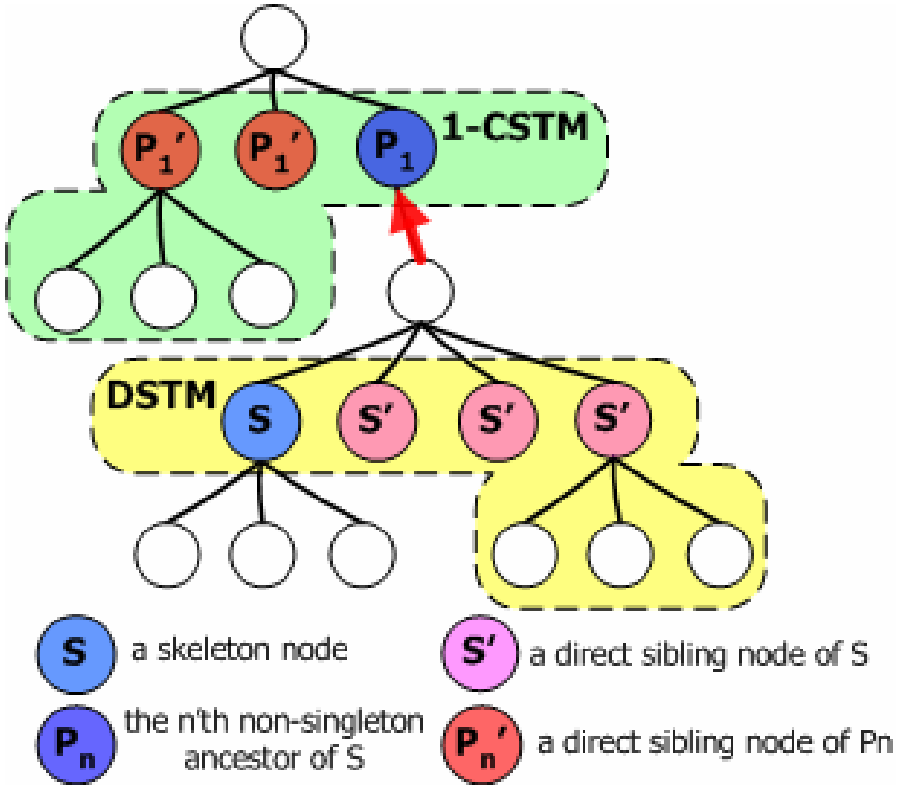
$$SII(N, f_i) = -\sum_{j=0}^{m-1} w_{ij} \log_m w_{ij} \qquad w_{ij} = \frac{f_i(n_j)}{\sum_{k=0\sim m-1} f_i(n_k)}, \forall n_k \in children(N).$$

WISDOM – 2. etap, uwagi

- przy wyszukiwaniu kandydatów na bloki brane są pod uwagę dodatkowe heurystyki:
  - średnia ważkość słów w blokach z artykułami musi być dostatecznie duża
  - średnie API odnośników w blokach TOC musi być dostatecznie duże
- wartość ST (SII Threshold) jest ustalana odgórnie i pozwala regulować jakość/liczbę zwróconych bloków

WISDOM – 3. etap

- otrzymana w poprzedniej fazie szkieletowa struktura IS poddawana jest dwóm operacjom łączenia poddrzew:
  - DSTM (Direct Tree Sibling Merging)
  - CSTM (Collateral Sibling Tree Merging)
- operacje te pozwalają na scalenie elementów należących do tego samego obiektu (tytuł artykułu, jego data, autor etc.)

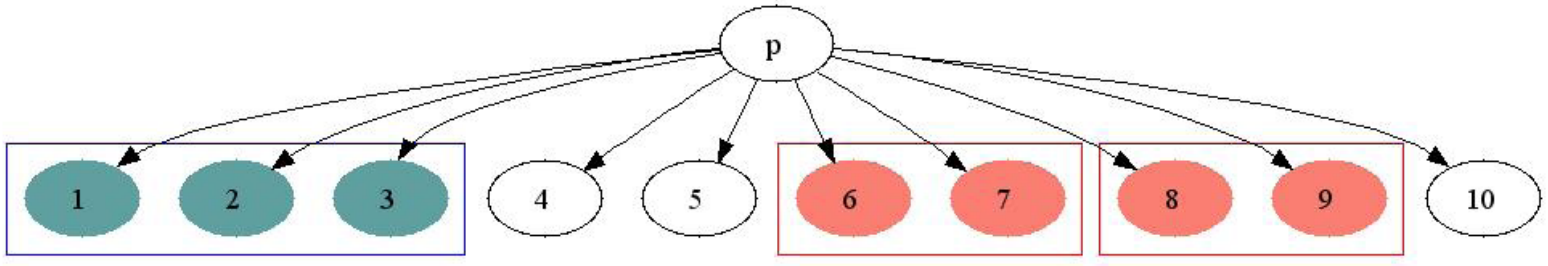


IKM – Intelligent Knowledge Mining

- korzystający z rozwiązań systemu MDR, zaproponowanego parę lat wcześniej
- przeznaczony do wykrywania fragmentów stron WWW zawierających listy obiektów podobnych (rekordów)
- podczas pojedynczego przebiegu algorytmu przetwarzana jest tylko jedna strona WWW
- rezultatem działania jest lista bloków zawierających rekordy (oraz wskazanie na obszar, w którym te bloki się znajdują) z przypisanymi wartościami istotności

IKM – ogólna zasada działania

- w oparciu o wczytaną stronę WWW budowana jest struktura DOM, a następnie znajdowane są bloki tzw. wzorców regularnych
- dla każdego bloku badane są 3 cechy:
  - odsetek obszaru zajmowanego przez grupę wzorców w ramach nadrzędnego bloku
  - stopnień podobieństwa wzorców regularnych wchodzących w skład jednej grupy wzorców
  - stopień wewnętrznego zróżnicowania wzorców wchodzących w skład grupy
- istotność grupy wzorców jest iloczynem tych 3 cech



IKM – ustalanie podobieństwa obiektów w ramach jednej grupy

- dla każdej grupy obliczamy stopień regularności tożsamy parametrowi *RPgroup*
- e - znormalizowana odległość Levenshteina pomiędzy sąsiednimi wzorcami regularnymi
- m - pomniejszona o 1 liczba wzorców regularnych w grupie

$$(RPgroup_k(N)) = 1 - \sum_{i=1}^{m} \frac{e_i}{m}$$

IKM – określanie zróżnicowania w obrębie pojedynczego obiektu w grupie

- zróżnicowanie wewnątrz pojedynczego wzorca regularnego badane jest przez porównywanie IS (Item Styles)
- IS jest to konkatenacja nazw znaczników łączących korzeń wzorca z wybranym liściem (włączając w to nazwy korzenia i liścia)

$$IS_1 : TD - TR - TBODY$$
$$IS_2 : IMG - A - TD - TR - TBODY$$
$$IS_3 : A - B - TD - TR - TBODY$$
$$IS_4 : INPUT - FROM - TD - TR - TBODY$$
$$IS_5 : P - TD - TR - TBODY$$
$$IS_6 : FONT - B - TD - TR - TBODY$$
$$IS_7 : B - FONT - TD - TR - TBODY$$
$$IS_8 : B - TD - TR - TBODY$$
$$IS_9 : A - CENTER - TD - TR - TBODY$$
$$IS_{10} : A - CENTER - TD - TR - TBODY$$

$$EN(IS_i) = -\sum_{j=1}^{m} w_{ij} \log_m w_{ij},\ w_{ij} \geq 1 \text{ and } m = |S|,\ S \text{ is the set of items style of an object}$$

$$where\ w_{ij} = \frac{f_{match}(IS_i, IS_j)}{the\ number\ of\ IS\ equal\ IS_i\ in\ the\ same\ object}$$

$$f_{match}(IS_i, IS_j) = \begin{cases} 1, if\ IS_i = IS_j \\ 0, otherwise \end{cases}$$

$$Diversity(RPgroup) = \frac{\sum_{i=1}^{n} W_{RP_i}}{n},\ where\ W_{RP_i} = \sum_{j=1}^{m} IS_j$$

Automatyczne znajdowanie szablonów

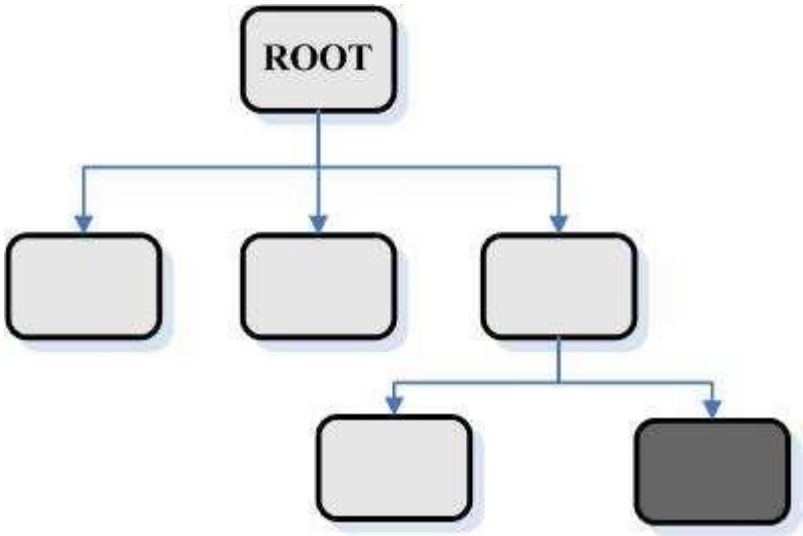
- powstał z myślą o dużych wyszukiwarkach internetowych
- przeznaczony do wykrywania powtarzających się fragmentów stron WWW należących do jednej witryny
- podczas pojedynczego przebiegu algorytmu przetwarzane są strony należące do tej samej witryny
- rezultatem działania jest zbiór stron WWW pozbawiony elementów wspólnych

Szablony - etapy algorytmu

1) utworzenie drzewa znaczników i pogrupowanie bloków wg ich cech i pozycji w drzewie
2) utworzenie odwrotnego indeksu i wykrycie na jego podstawie szablonów

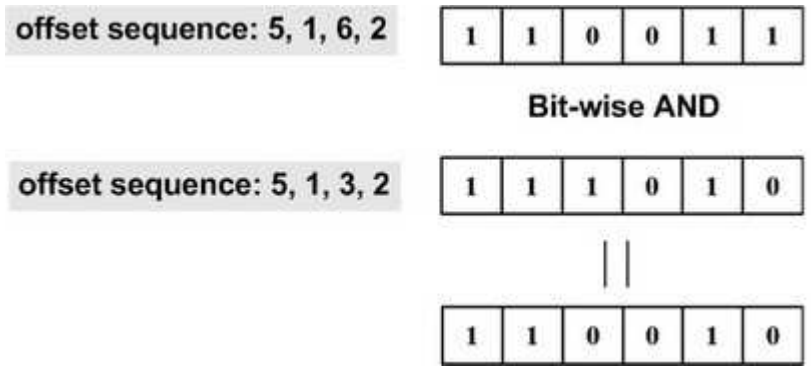
Szablony – 1. etap

- dla każdej wczytanej strony tworzone jest drzewo znaczników z pominięciem niektórych z nich, np. <TD> i <TR>
- każdemu blokowi przypisywana jest pozycja w strukturze
- bloki ze wszystkich wczytanych stron są grupowane
- otrzymane w ten sposób bloki BSC (Block Style Cluster) odznaczają się tym, że dla każdej pary bloków do nich należących pozycje i wartości atrybutów znaczników bloków pokrywają się



Szablony – 2. etap

- każda strona poddawana jest indeksowaniu odwrotnemu, tzn. każdemu napotkanemu słowu przypisana jest lista par składających się z identyfikatorów bloków i binarnych list wystąpień
- po zakończeniu procesu indeksowania dla każdego słowa realizowane jest grupowanie przypisanych mu par w celu utworzenia WFC (Word-Feature Cluster)
- każdy WFC składa się z bloków takich, że:
  - należą one do tego samego BSC
  - ich sekwencje położeń są do siebie podobne
- binarna lista wystąpień jest to ciąg składający się z 32 bitów
- ustalenie podobieństwa 2 list odbywa się poprzez zliczenie jedynek w sekwencji będącej rezultatem wykonania operacji AND i przyrównanie do z góry zdefiniowanego progu



Szablony – 2. etap c.d.

- jeżeli dla danego słowa w indeksie występuje dostatecznie liczny WFC, to słowo to jest uznawane za „szablonowe”
- jeśli blok posiada duży odsetek słów „szablonowych”, to jest uznawany za element szablonu strony i zostaje z niej usunięty

Podsumowanie

- metody w pełni automatyczne dają bardzo zadowalające wyniki, zarówno jeśli chodzi o jakość działania (precision, recall, f-measure), jak i szybkość
- warto więc dążyć do utworzenia narzędzia pozwalającego na automatyczne rozpoznawanie możliwie najwięcej różnych klas obiektów na stronach WWW

Literatura

- **A Comparative Study on Classifying the Functions of Web Page Blocks**. Xiangye Xiao, Qiong Luo, Xing Xie, Wei-Ying Ma. 2006.
- **The Mining and Extraction of Primary Informative Blocks and Data Objects from Systematic Web Pages**. Yi-Feng Tseng, Hung-Yu Kao. 2006.
- **Mining Data Records in Web Pages**. Bing Liu, Robert Grossman, Yanhong Zhai. 2003.
- **Template Detection for Large Scale Search Engines**. Liang Chen, Shaozhi Ye, Xing Li. 2006.
- **WISDOM: Web Intra-page Informative Structure Mining based on Document Object Model**. Hung-Yu Kao, Jan-Ming Ho, Ming-Syan Chen. 2005.

DZIĘKUJĘ ZA UWAGĘ!