Wyszukiwanie i Przetwarzanie Informacji WWW

Automatyczne zbieranie dokumentów WWW

1: Podstawy

Marcin Sydow

PJWSTK

Plan dzisiejszego wykładu:

Automatyczne Zbieranie Kolekcji Dokumentów WWW.

- Wprowadzenie
- Zagadnienia Koncepcyjne
- Etykieta
- Zagadnienia Techniczne
- Podsumowanie materiału

Nazewnictwo

Omawiany w niniejszym wykładzie moduł zbierający ma wiele równolegle istniejących nazw (szczególnie w jęz. angielskim).

Jako jednowyrazowa nazwa polska, w tym wykładzie, będzie używana nazwa **zbieracz**. Istnieją też inne określenia: robot, bot, albo szczególnie specyficzne: pająk

W języku angielskim używa się nazw: **crawler**, Web bot/robot, spider, wanderer

Do czego służy Moduł Zbierania? (ang. crawler)

W klasycznym IR, kolekcje były dostarczane bezpośrednio przez operatorów/użytkowników systemów.

W przypadku WWW nie ma oczywiście katalogu wszystkich dostępnych dokumentów WWW.

Wyszukiwarka musi zapewnić bazową kolekcję dokumentów w ramach swoich normalnych operacji - do tego służy Moduł Zbierania (ang. crawler)

Moduł Zbierania - zasada działania

Startowy zestaw adresów URL - **inicjalizuje** kolejkę

Dla każdego URL:

1. ściągnąć
2. wyparsować następne adresy URL
3. dorzucić je do kolejki

Postępować tak, aż do zajdą **warunki zakończenia** (np. wyczerpanie zasobów)

Następnie (niezależnie od procesu zbierania):

1. ściągnięte dokumenty trafiają do **Repozytorium**
2. są później podawane do Modułu Indeksującego - powstaje **Indeks**

Pozostałe Fazy

Dokument po ściągnięciu jest na ogół

- kompresowany (pojedynczo lub w porcji, razem z innymi dokumentami)
- parsowany w celu wydobycia kolejnych adresów URL (hiperlinków wychodzących z danego dokumentu) do umieszczenia w kolejce zadań
- zapisywany w repozytorium na dysku

Zarządca kolejki zapewnia odpowiednią ilość bieżących zadań do ściągnięcia tak aby optymalnie wykorzystać interfejs sieciowy i zarazem przestrzegać szeregu **strategii** zbierania

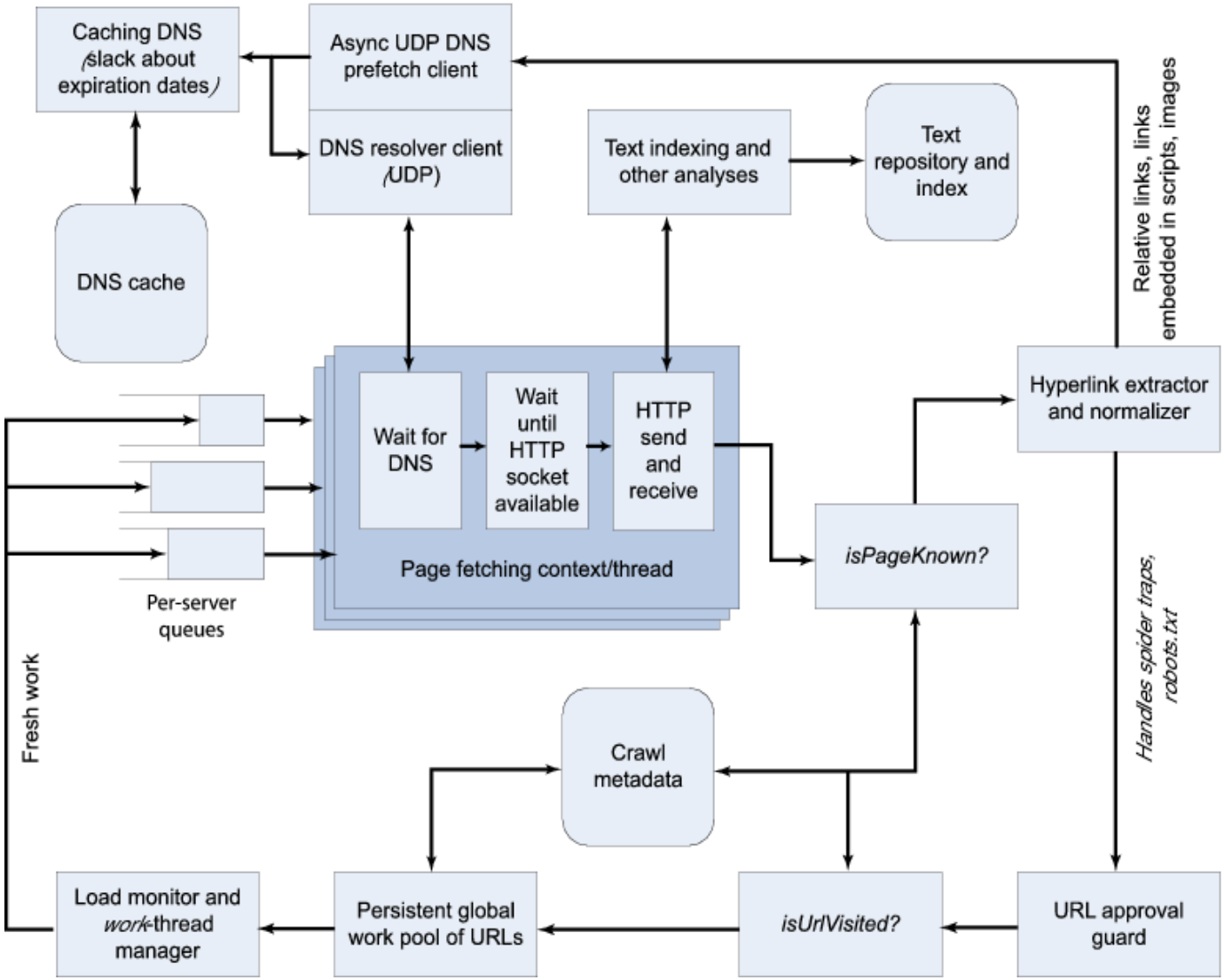
Modyfikacje Podstawowego Zadania

Istnieją różne warianty podstawowego zachowania Modułu Zbierania, w zależności od jego przeznaczenia.

Przykłady:

- zbieranie **maksymalnej liczby dokumentów** z określonej grupy hostów lub domen
- zbieranie dokumentów z jak **największej liczby hostów** w danym uniwersum (np. domena .pl)
- zbieranie **"tematyczne"** (ang. focused crawling)

Architektura Zbieracza



(wg. "Mining the Web" S.Chakrabarti, Morgan-Kaufmann, 2003)

Sterowanie Zbieraniem

Do sterowania procesem zbierania moduł zbierający może używać informacji z poprzedniego cyklu zbierania lub z innych modułów systemu wyszukiwawczego

Do informacji takich należą np.

- struktura linków
- wielkość hosta/domeny
- jakość dokumentów/hosta/domeny
- tematyka dokumentów/hosta/domeny
- w jakim stopniu dany host(domena) okazał się chłamem (ang. spam)
- statystyki zapytań
- statystyki ruchu WWW

Podstawowe Problemy

Moduł zbierający **musi** stawić czoła następującym problemom:

- Odporność (ang. robustness):
  - pułapki
  - chłam wyszukiwarkowy (ang. search engine spam)
  - błędy
  - awarie sprzętu
  - wyczerpanie zasobów
- **Etykieta** - ograniczenia:
  - jawne (robots.txt, etc.)
  - niejawne (szanowanie cudzych zasobów - przepustowości, CPU, etc.)

Pożądane Cechy:

Oprócz tego, dobry moduł zbierający powinien mieć następujące cechy:

- rozproszenie
- skalowalność
- maksymalnie wydajne użycie własnych zasobów (CPU, RAM, dysk, sieć, etc.)
- zapewnienie jakości zbieranej kolekcji
- zapewnienie świeżości zbieranej kolekcji
- rozszerzalność

Strategie Zbierania

Przy projektowaniu i sterowaniu modułem zbierającym należy też ustalić kilka **strategii**:

- Które dokumenty zbierać?
- Jak odświeżać dokumenty?
- Jak zminimalizować obciążenie cudzych zasobów?
- Jak uwspółbieżnić proces zbierania?

Które dokumenty zbierać?

Niemożliwe jest zebranie wszystkich dobrych dokumentów, ze względu na skalę sieci WWW.

Dodatkowo:

- istnienie dokumentów dynamicznych czy aktywnych
- ciągła ewolucja sieci WWW
- "ukryty Web" (ang. "hidden Web")

sprawiają, że można powiedzieć, iż **WWW jest nieskończony**

Dlatego należy obrać **strategię wyboru dokumentów** do zbierania

Strategia wyboru dokumentów

Dotyczy zapewnienia zbierania "ważnych" dokumentów w pierwszej kolejności

Należy tu rozważyć kilka aspektów:

- jak zdefiniować miarę "ważności" strony
- jaki wybrać algorytm priorytetowania w oparciu o w/w miarę
- jak "zgadywać", które następne strony są "ważne" - bez możliwości uprzednego sprawdzenia ich zawartości

Miary ważności strony

Jakie strony uznać za ważne? Przykłady:

- strony o "interesującej" **zawartości** (np. zawierające określone słowa kluczowe, będące w określonym sąsiedztwie tematycznym, zawierające dużo obrazków, etc.)
- strony **"popularne"** w sensie struktury linków lub ruchu internetowego (np. mające wysoką wartość PageRank lub innych miar opartych na analize linków grafu WWW)
- strony o określonej **lokalizacji** (np. strony w domenie .pl, albo zawierające człon "edu")

Protokoły Etykiety

Autorzy niektórych zasobów WWW mogą zdecydować, iż **życzą sobie** (albo jest to konieczne z innych względów - np. prawnych) aby **nie były one ściągane** lub odwiedzane przez systemy zbierające.

Niepisana etykieta dotycząca zachowania zbieraczy zakłada, iż **przestrzegają** one poniższych **protokołów etykiety** (a przynajmniej **pierwszego**):

- tzw "Protokół Wykluczania Robotów" (ang. The Robots Exclusion Protocol)
- tzw "Protokół Meta" dotyczący zbierania (ang. The Robots META tag)

Protokoły te mogą być jedynie **dobrowolnie** przestrzegane przez roboty. Jeśli to nie skutkuje można zastosować silniejsze środki ochrony (np. hasło, albo techniki rozpoznania człowieka coraz bliższe idei "testu Turinga")

Robots Exclusion Protocol

W pliku `robots.txt`, na ogół w katalogu głównym danego portalu (ang. site), umieszczone są informacje dotyczące zasad postępowania zbieracza.

Pojedynczy rekord ma pierwsze pole `User-agent:` (jego wartość określa nazwę zbieracza, albo symbol '*' dotyczący wszystkich zbieraczy) oraz listę pól `Disallow:` (jego wartość określa nazwę katalogu lub pliku, który nie ma być zbierany). Pola są w oddzielnych liniach, rekordy są oddzielane pustymi liniami.

Pusty plik robots.txt albo pusta wartość pola `Disallow:` określa, że wszystko jest dostępne. Plik musi być dostępny przez protokół HTTP.

Informacje na temat protokołu są dostępne pod adresem:
`www.robotstxt.org`

Przykład

```
User-agent: *
Disallow: /draft.html
Disallow: /temp/
Disallow: /fotki/robocze/

User-agent: crawl.pl
Disallow:
```

Przykład

```
User-agent: *
Disallow: /draft.html
Disallow: /temp/
Disallow: /fotki/robocze/

User-agent: crawl.pl
Disallow:
```

Powyższy zapis zabrania w katalogu głównym, wszystkim zbieraczom ściągania dokumentu `draft.hml`, oraz wszystkich z katalogów `temp` oraz `fotki/robocze` **za wyjątkiem** zbieracza o nazwie `crawl.pl`

Rozszerzenia

Zaproponowano pewne **rozszerzenia** składni pliku `robots.txt`. Należą do nich m.in. pola: `Allow:`, `Request-rate:` i `Visit-time:`

Praktyczna implementacja systemu zbierającego, który miałby stosować się do wszystkich proponowanych rozszerzeń nie jest jednak łatwa (szczególnie dla zbieraczy średniej i dużej skali).

Zbieracze poszczególnych wyszukiwarek czasami obsługują niektóre rozszerzenia (np. pole `Allow:` czy dodatkowe symbole wieloznaczne (np. '*' czy '$').

Informacje takie są dostępne na stronach podmiotów odpowiednich dla danych zbieraczy.

Protokół wykluczania dla autorów dokumentów

Plik `robots.txt` może być umieszczony tylko **jednokrotnie** dla danego portalu (ang. site). Umieszczanie dodatkowych plików `robots.txt` w podkatalogach jest przez zbieracze **ignorowane**. Do zarządzania takim plikiem na ogół uprawniony jest tylko administrator portalu.

Dlatego też, dla wygody autorów dokumentów html, istnieje **dodatkowy protokół** (ang. "The META tag protocol").

W każdym dokumencie html można umieścić specjalny znacznik `meta` określający czy zawartość danego dokumentu może być ściągana oraz czy można używać (analizować) wychodzące z niej hiper-linki.

Użycie

W każdym nagłówku dokumentu html można umieścić znacznik `meta` z pierwszym atrybutem `name=''robots''` i następnym `content=''<lista dyrektyw>''`, gdzie obecnie zdefiniowano 4 dyrektywy: `index`,`noindex`,`follow`,`nofollow`.

Przykład:

```
<html>
<head>
<meta name="robots" content="noindex,nofollow">
</head>
...
```

Stosowanie się do tego protokołu przez systemy zbierające nie jest tak powszechne jak do protokołu "`robots.txt`"

Parsowanie linków i Normalizacja

Zwykle w zbieraczach dużej skali stosuje się specjalne parsery do wydajnego i odpornego na błędy parsowania linków (flex)

Poważnym zagadnieniem jest **normalizacja** adresów URL. Różne, bowiem, adresy URL mogą odnosić się do tego samego dokumentu z następujących powodów:

- w praktyce relacja `IP - nazwa hosta` jest relacją "wiele do wielu"
- kodowanie (np. base-64)
- kapitalizacja
- używanie ścieżek względnych (np "../../", etc.)
- domyślność/opcjonalność niektórych fragmentów URL (np. nazwa protokołu, numer portu, przyrostek ".html", plik "index.html", zapytanie , fragment dokumentu, etc.)

Wszystkie powyższe niejednoznaczności sprawiają, że reguły normalizacyjne (i ich obliczanie) są dość skomplikowane

Unikanie wielokrotności

Typowo, w kolejce zbierania trzyma się tylko unikatowe adresy URL. Z tego powodu, zanim nowy URL zostanie dodany do kolejki należy sprawdzić czy jeszcze go tam nie ma.

Operacja ta odbywa się w praktyce bardzo często, więc powinna być bardzo szybka. Osiąga się to zwykle obliczając kod URL (można stosować np. algorytm MD5) i stosując tablice mieszające (trzymane w pamięci RAM).

Zaleca się 2-członową strukturę kodu mieszającego (np. dla hosta i dla "reszty" adresu URL), aby wykorzystać występującą w WWW **lokalność** odwołań (przeciętnie odwołania wewnątrz hosta są dużo częstsze niż poza dany host)

Unikanie Duplikatów

Mimo stosowania normalizacji adresów URL, problemem o istotnym znaczeniu jest **unikanie duplikatów**, tzn. wielokrotnego ściągania dokumentów o **takiej samej zawartości**, ponieważ dokumenty o różnych adresach URL mogą mieć bardzo podobną (lub identyczną) treść. Powodem jest m.in. replikacja zawartości niektórych portali (ang. mirroring).

Należy zauważyć, że ściągnięcie duplikatu dokumentu może oznaczać nie tylko zbędne zużycie zasobów (zbieracza, dysków, etc.), ale także **wielokrotne dodanie adresu URL** do kolejki, jeśli jest on zapisany w sposób względny.

Techniki wykrywania identycznej treści

Wydajne wykrycie dokładnych duplikatów jest łatwe: można obliczyć np. sumę kontrolną zawartości dokumentu (np. MD5).

W celu unikania wielokrotnego dodawania do kolejki linków zapisanych względnie (z dwóch identycznych dokumentów d_1 i d_2) można też dla względnych adresów URL postaci $d_1/link$, $d_2/link$ reprezentować przedrostki za pomocą haszowania ich treści $h(d_1)/link$, $h(d_2)/link$ - pozwoli to uniknąć wielokrotnego dodania tego samego linku **o ile dokumenty są identyczne**.

W praktyce jednak, problemem są dokumenty **prawie identyczne**, różniące się jedynie drobnymi szczegółami (np. datą modyfikacji, adresem administratora).

Dla dokumentów takich sumy kontrolne będą oczywiście różne.

Wykrywanie dokumentów prawie identycznych

Problem polega na stwierdzeniu, czy dwa dokumenty (tekstowe) mają **podobną** treść.

Istnieje wiele **miar odległości edycyjnych** (np. Levenshtein, Monge-Elkan, Soundex). Niska wartość takiej miary dla dwóch łańcuchów tekstowych, oznacza że łańcuchy te są **podobne**. Niestety większość takich miar jest zbyt **intensywna obliczeniowo** (np. o złożoności kwadratowej) aby stosować je do treści całych dokumentów WWW (średnio ok. 10KB) w procesie ściągania.

Popularną techniką o niższej złożoności obliczeniowej, praktycznie stosowalną do wykrywania dokumentów bardzo podobnych jest technika oparta na `q-gramach` (ang. **shingling**).

Technika ta jest też użyteczna w ostatniej fazie wyszukiwania - prezentacji wyników zapytania - do grupowania wyników bardzo podobnych.

Pułapki

W praktyce, duża część dokumentów obecnych w WWW zawiera błędy składniowe (i nie tylko). System zbierający musi być na nie odporny.

Ciekawym zjawiskiem socjologicznym są tzw. **pułapki**, czyli celowo sporządzone dokumenty lub całe grupy dokumentów mające doprowadzić do załamania systemu zbierającego.

S.Chakrabarti ("Mining the Web") cytuje natknięcie się na URL z 68 tysiącami znaków NULL w środku (!), który spowodował załamanie się nawet specjalnego dedykowanego parsera przygotowanego za pomocą narzędzia `flex`.

Pułapki, c.d.

Innym rodzajem pułapek są dynamiczne strony produkujące "nieskończone" kolekcje dokumentów.

Ostatnio ważnym zjawiskiem jest tzw. **chłam wyszukiwarkowy** (spam), na który system zbierający musi być specjalnie przygotowany.

Na ogół nie "ściąga się" plików, które nie są parsowalnymi dokumentami (np. skrypty CGI albo dokumenty aktywne), co częściowo eliminuje część zagrożeń

Monitorowanie

W praktyce, automatyczne zbieranie dokumentów WWW wymaga **systematycznego monitorowania**. Dotyczy to m.in. **automatycznego** śledzenia statystyk związanych z:

- przepustowością sieciową
- częstością ściągania dokumentów
- zużyciem zasobów (RAM, CPU, dyski)

Oprócz tego, w praktyce, **konieczne** jest umożliwienie przesyłania informacji, uwag i komentarzy od administratorów odwiedzanych portali. Zwykle umieszcza się nazwę systemu zbierającego **wraz z kontaktowym adresem e-mail** lub numerem telefonicznym w nagłówku żądania HTTP wysyłanego przez zbieracza i **powołuje** osobę (lub grupę) odpowiedzialną za natychmiastowe reagowanie na wszelkie sygnały.

Na zaliczenie tego wykładu:

1. podstawowy cykl pracy zbieracza
2. konieczne i pożądane cechy zbieracza
3. zagadnienia strategii zbierania
4. na czym polega “etykieta zbierania”
5. robots exclusion protocol
6. dlaczego należy podać e-mail w nagłówku wysyłanym przez zbieracz
7. jak uniknąć duplikatów dokumentów
8. pułapki
9. dlaczego należy monitorować proces zbierania

Dziękuję za uwagę

Dziękuję za uwagę.