# Wyszukiwanie i Przetwarzanie Informacji WWW

## Wyszukiwanie ogólnie i w korpusach tekstowych. "Boolowskie" IR, przetwarzanie tekstu i zapytań

Marcin Sydow

Web Mining Lab, PJWSTK

# Plan dzisiejszego wykładu:

- Wprowadzenie
- Przygotowanie tekstu
- Zapytania
- Podsumowanie Wykładu

# Cyrkulacja informacji

Najbardziej ogólnie można powiedzieć, że informacja/wiedza jest zapisywana i odczytywana przez **agentów** (ludzi lub maszyny) w postaci **dokumentów** do i z pewnych **repozytoriów**, za pomocą pewnych **systemów pośredniczących** np.:

- książki (czytanie i pisanie)
- biblioteki (zamawianie i pożyczanie)
- umysł człowieka (rozmowa, systemy typu Yahoo! Answers)
- WWW (tworzenie i przeglądanie stron)
- kolekcje dokumentów tekstowych (“klasyczne” systemy IR)
- relacyjne bazy danych

# Potrzeba informacyjna

Zakładamy, że powodem wyszukiwania w repozytorium jest **potrzeba informacyjna** - abstrakcyjne pojęcie pierwotne np.

- chęć poznania jakiejś epoki (np. wypożyczanie odpowiedniej książki z biblioteki)
- sprawdzenie kiedy odchodzi autobus (np. za pomocą wyszukiwarki WWW)
- dowiedzenie się jaka jest najwyższa góra w Kolumbii (np. systemy typu QA)
- podliczenie miesięcznych wydatków na reklamę w firmie (np. relacyjna baza danych)
- automatyczne dopasowanie reklamy wyszukiwarkowej do wyników

# Zapytania

Aby wyrazić **potrzebę informacyjną** potrzebny jest pewien **język zapytań**, który specyfikuje jaka jest składnia zapytań (syntaktyka) i jaką mają one interpretację (semantyka), np.:

- język naturalny (zapytanie między ludźmi)
- formularz (np. biblioteka)
- ciąg słów kluczowych (wyszukiwarka WWW)
- zapytanie SQL (relacyjna baza danych)
- wyrażenie regularne (np. wyszukiwanie grepem w logach)
- obrazek (“znajdź podobny”)

# Kryteria języków zapytań

- precyzja (np. wyrażenia regularne)
- ekspresyjność (np. język naturalny)
- stopień złożoności (np. SQL)
- prostota (np. interfejs wyszukiwarek WWW)

Oczywiście, zwykle mamy do czynienia z pewnym kompromisem pomiędzy prostotą a ekspresyjnością lub precyzją, etc.

# System IR - sformułowanie

Bardziej formalnie, system IR zakłada następujące elementy:

- potrzeba informacyjna
- zapytanie (sformułowane w języku zapytań) - (przybliżona) reprezentacja potrzeby informacyjnej
- korpus dokumentów (przybliżona reprezentacja wiedzy)
- (przybliżona) reprezentacja dokumentów
- prezentacja wyników

System ma za zadanie **zaspokoić potrzebę informacyjną** w opaciu o zapytanie i korpus za pośrednictwem (niedoskonałych) reprezentacji. Wynik jest **prezentowany** w jakiejś formie (np. zbiór dokumentów tekstowych, uporządkowana lista hyperlinków do stron WWW, etc.)

# Wyszukiwanie Informacji

- Jak widzimy jest to termin o bardzo szerokim znaczeniu
- Dziedzina niemłoda, o sporej tradycji (ang. Information Retrieval) (conajmniej od lat 60-tych XX wieku)
- W tym kursie skupimy się głównie na wyszukiwaniu w kolekcjach tekstowych i w WWW
- Istotnym wyróżnikiem omawianych dziedzin będzie
  - brak precyzyjnie zdefiniowanej struktury dokumentu
  - zaszumienie dokumentów
  - nieprecyzyjne zapytania

Tak więc, omówienie nie dotyczy np. relacyjnych baz danych, gdzie zarówno dane jak i zapytania mają bardzo precyzyjnie zdefiniowaną strukturę

# Klasyczny tekstowy system IR (tzw. "boolowski")

Zasada działania:

- Mamy korpus dokumentów tekstowych D.
- Mamy zapytanie boole'owskie q traktowane jako zbiór albo lista słów kluczowych.
- System ma zwrócić dokumenty z D odpowiadające zapytaniu q.

Istotne jest to, że zwraca się **wszystkie i tylko te** dokumenty, które **dokładnie** pasują do zapytania. Stąd nazwa "boole'owskie".

# Założenia w klasycznym IR

Zakłada się:

- wysoką jakość tekstów w korpusie (przygotowane przez ludzi)
- brak zaszumienia i jednorodność dokumentów (język, rozmiar, format, etc.)
- brak czynnika "wrogości"

Założenia te są istotne dla modelu - warunkują metody wyszukiwania. Nie są one spełnione np. w WWW (WIR).

# Odrobina historii

Systemy IR są starsze niż wyszukiwarki WWW, ale posłużyły za podstawę ich pierwszej generacji:

- kontrolowane kolekcje dokumentów w urzędach i korporacjach
- Archie (przeszukiwanie FTP)
- Alta Vista (połowa lat 90.)

Web: dodatkowe informacje (linki, znaczniki, etc.) ... i dodatkowe problemy... (np. spam)

# (Pre)historia WIR w skrócie

- 1611: prototyp indeksu (Strong's Exhaustive Concordance of Bible)
- 1945: Memex - "prototyp" WWW (V.Bush "As we may think")
- 1960: SMART Information Retrieval System (G.Salton, Cornell Univ.)
- 1965: Xanadu - *hypertext* (Ted Neson)
- 1980: system do nawigacji po dokumentach (T.Berners-Lee)
- 1990: narodziny WWW (Tim Berners-Lee, CERN)
- 1993-95: pierwsze przeglądarki (Mosaic/Netscape)
- 1994: Lycos - pierwsza wyszukiwarka
- 1994: WebCrawler, 4K hostów (Brian Pinkerton)
- 1994: "Jerry's Guide to the World Wide Web" (później: Yahoo)
- 1995: AltaVista, Excite, InfoSeek, Inktomi
- 1996: Yahoo wchodzi na giełdę
- 1996-1998: początki Google

# Klasyczny system IR - zasada działania

Zanim system IR będzie gotowy do pracy, musi przygotować tzw. **"odwrócony indeks"** (czyli główną strukturę danych typu słownikowego, informującą w których dokumentach występują dane słowa kluczowe)

W odpowiedzi na zapytanie q:

1. reprezentuje q w postaci wygodnej do obliczeń
2. wynajduje wszystkie dokumenty spełniające warunki q (w tym celu konsultuje indeks)
3. prezentuje wyniki

# Klasyczny system IR - przygotowanie indeksu

Fazy przygotowawcze systemu IR:

- oczyszcza kolekcję dokumentów
- wybiera tokeny do indeksowania
- tworzy odwrócony indeks
- (zazwyczaj) kompresuje indeks

# Przygotowanie kolekcji dokumentów (ang. preprocessing)

- Ustalenie co rozumiemy przez pojedynczy *token*
- Oczyszczanie. Usunięcie interpunkcji, znaczników (np. html), pewnych znaków (np. niealfanumerycznych, cyfr)
- sprowadzenie do małych liter
- sprowadzenie do formy podstawowej (kury, kurę -> kura)
- zwinięcie końcówek (ang. stemming) np. kura -> kur,
- tokeny -> liczby całkowite (32b wystarczy)

Na ogół zapytanie poddaje się **dokładnie takim samym operacjom**.

# Niby-wyrazy (ang. stopwords)

Ich udział w kolekcji jest bardzo wysoki, ale mają niską wartość dyskryminacyjną:

a, aby, ale, i, oraz, ten, ...

Nie indeksuje się ich. Można natomiast zapamiętać miejsca, gdzie były. Dzięki temu wciąż można szukać fraz, które je zawierają.

Przy wyrzucaniu niby-wyrazów trzeba być ostrożnym i nie "wylać dziecka z kąpielą". Np. następująca ważna sentencja jest niczym innym jak listą niby-wyrazów:

# Niby-wyrazy (ang. stopwords)

Ich udział w kolekcji jest bardzo wysoki, ale mają niską wartość dyskryminacyjną:

a, aby, ale, i, oraz, ten, ...

Nie indeksuje się ich. Można natomiast zapamiętać miejsca, gdzie były. Dzięki temu wciąż można szukać fraz, które je zawierają.

Przy wyrzucaniu niby-wyrazów trzeba być ostrożnym i nie "wylać dziecka z kąpielą". Np. następująca ważna sentencja jest niczym innym jak listą niby-wyrazów:

To be or not to be...

# Niby-wyrazy (ang. stopwords)

Ich udział w kolekcji jest bardzo wysoki, ale mają niską wartość dyskryminacyjną:

a, aby, ale, i, oraz, ten, ...

Nie indeksuje się ich. Można natomiast zapamiętać miejsca, gdzie były. Dzięki temu wciąż można szukać fraz, które je zawierają.

Przy wyrzucaniu niby-wyrazów trzeba być ostrożnym i nie "wylać dziecka z kąpielą". Np. następująca ważna sentencja jest niczym innym jak listą niby-wyrazów:

To be or not to be...

Trzeba też uważać na wieloznaczność - wyrazy "jak", "je", "go" są niby-wyrazami, ale jednocześnie (w innym znaczeniu) są wyrazami niosącymi informację.

# NLP: polski vs angielski

- Dla języka angielskiego badania NLP są zaawansowane. Jest też dostępnych sporo narzędzi (w szczególności klasyczny algorytm Portera – stemming), WordNet, etc.
- Badania NLP dla języka polskiego są mniej zaawansowane i jest na nie mniej środków (interesuje głównie Polaków, w przeciwieństwie do języka angielskiego)
- NLP dla polskiego jest bardziej skomplikowany niż dla angielskiego. Decyduje o tym fleksja i bogatsza gramatyka - bliższa klasycznej Łacinie: koniugacje, deklinacje, 7 przypadków, rodzaje (nota bene: jest ich ponad 3)
- uwaga: przy rozpoznawaniu form zbiór odpowiedzi może się zbytnio rozszerzyć (zawierając niepożądane wyniki).

# Zapytania boole'owskie - przykłady

Zwróć dokumenty:

- zawierające słowo “Linux”
- zawierające słowo “Linux” ale nie zawierające słowa “Suse”
- zawierające słowo “POSIX” lub frazę “GNU Linux”
- w których słowa “GNU” i “ssak” występują w tym samym zdaniu

Dwa ostatnie to zapytania rozszerzone - używające pojęcia frazy lub bliskości (ang. proximity queries)

# Operatory Zapytań

W wyszukiwarkach opartych na modelu boole'owskim **domyślnym operatorem jest operator AND**. Tzn. zapytanie:

```
GNU Python
```

Dotyczy wszystkich dokumentów (zindeksowanych przez dany system) zawierających **równocześnie** wyraz "GNU" i wyraz "Python". Na ogół kapitalizacja jest ignorowana.

Oprócz tego, dostępne są na ogół pozostałe z podstawowych operatorów logicznych:

- NOT
- OR

# Przykłady: NOT, OR

Np. zapytanie:

`python NOT gnu`

Dotyczy dokumentów w systemie, które zawierają wyraz “python” **i nie zawierają** wyrazu “gnu”.

A zapytanie:

`python OR ruby`

Dotyczy dokumentów zawierających wyraz “python” lub zawierających wyraz “ruby” (lub oba).

# Operator Frazy

Do innych, ważnych operatorów zapytań standardowo należy dzisiaj operator frazy. Jest na ogół oznaczany przez znaki cudzysłowów otaczające grupę (lub) grupy wyrazów. Np.

"Programming Python"

Dotyczy dokumentów zawierających frazę "programming python" (wyrazy muszą wystąpić obok siebie). Zauważmy, że jest to zawężenie zapytania:

Programming Python

gdzie wyrazy te mogą wystąpić gdziekolwiek w dokumencie.

# Automatyczna Korekta

Część zapytań przekazywanych do systemów wyszukiwawczych może być **błędnie** wpisana. (przypadkowe literówki; niedokładna znajomość terminów, nazw, etc.)
W takich przypadkach system może w różnym stopniu reagować na takie sytuacje:

- zwracać dokumenty dokładnie odpowiadające zapytaniu (brak reakcji)
- zwracać dodatkowo dokumenty odpowiadające **automatycznie poprawionemu** zapytaniu
- j.w. ale tylko jeśli oryginalne zapytanie zwraca **mniej niż** ustalony próg wyników
- wyświetlać proponowaną korektę zapytania (łącznie z oryginalnymi wynikami)

Istnieje kilka podejść do obliczania sugerowanej korekty pisowni zapytania (ang. spelling correction)

# Podział

Przedewszystkim, obliczanie autokorekty może być:

- izolowane (każdy wyraz z zapytania analizowany jest oddzielnie)
- kontekstowe (zapytanie analizowane jest jako całość)

Drugie podejście jest oczywiście potężniejsze, ale równocześnie bardziej wymagające obliczeniowo. Przykład:
“`faktury dom wysłania`”
(żaden wyraz brany pojedynczo nie zawiera literówki, ale zapytanie to potraktowane całościowo wygląda na zawierające oczywistą literówkę)

# Ogólny mechanizm

W przypadku gdy system wykryje, iż zapytanie prawdopodobnie zawiera błąd pisowni (np. na podstawie liczby zwróconych wyników) na ogół:

- oblicza się ograniczony **zbiór kandydatów** na prawidłową (zamierzoną przez użytkownika) wersję zapytania
- spośród nich wybiera się **najlepszego**, wg pewnego kryterium, i stosuje do dalszych obliczeń (np. wyświetla jako sugerowaną korektę)

Najczęściej podstawowym kryterium doboru jest **bliskość edycyjna** - wybiera się spośród **"prawidłowych"** zapytań kandydata najbliższego (względem danej miary) do oryginalnego.

# Odległości edycyjne

Przy porównywaniu 2 łańcuchów uwzględnia się podstawowe **operacje edycyjne**:

- wstawienie znaku
- pominięcie (usunięcie) znaku
- zamiana znaku (na inny)
- zamiana sąsiadujących znaków (transpozycja)

Na ogół **odległość** pomiędzy dwoma łańcuchami jest niemalejącą funkcją ilości operacji edycyjnych niezbędnych do przekształcenia pierwszego na drugi.

# Odległość Levenshteina

Najbardziej znaną miarą odległości jest miara **Leventshteina** zliczająca po prostu w/w operacje edycyjne (za wyjątkiem transpozycji). Ma ona łatwą implementację za pomocą algorytmu dynamiczego o złożoności rzędu $O(l_1 \cdot l_2)$, gdzie $l_1$ i $l_2$ to długości porównywanych łańcuchów.

Naturalnym rozszerzeniem jest zróżnicowanie kosztów poszczególnych kosztów operacji (Needleman-Wunsch) albo wręcz uzależnienie ich od modelu języka. Można też rozszerzyć zbiór dozwolonych operacji na tekście (np. Smith-Waterman).

# Inne Miary

Przykładowe miary stosowane w praktyce:

- Needleman-Wunsch (koszty zamian poszczególnych symboli)
- Smith-Waterman (wprowadza tzw. 'przerwy')
- Jaro (ilość i kolejność wspólnych znaków)
- Jaro-Winkler (dodatkowo długość wspólnego prefiksu)
- Monge-Elkan (wiele tokenów)
- oparta na q-gramach (bardzo szybka)
- Soundex
- NCS (ang. normalised compression distance)

# Lektury

Uzupełnić wiedzę można np. w poniższych publikacjach:

Podstawy IR są opisane w klasycznych pozycjach:

- G.Salton et al. "Introduction to Modern Information Retrieval", McGraw-Hill, 1983
- W.B. Frakes, R. Baeza-Yates "Information Retrieval: Data Structures and Algorithms", Prentice Hall, 1992

Tworzenie i kompresję indeksu opisano w książce:

- I.H. Witten, A. Moffat, T.C. Bell "Managing Gigabytes: Compressing and Indexing Documents and Images", Morgan Kaufmann, 1999

Ponieważ materiał tej prezentacji jest podstawowy, nie wymienia się tutaj specjalistycznych publikacji naukowych.

# Na zaliczenie tego wykładu:

1. ogólne sformułowanie systemu IR
2. przykłady problemów wyszukiwania
3. porównać dwa różne języki zapytań
4. klasyczny "boolowski" system IR
5. założenia klasycznego IR
6. przygotowanie kolekcji dokumentów (fazy)
7. operatory zapytania
8. autokorekta
9. odległość Levenshteina

# Dziękuję za uwagę

Dziękuję za uwagę.