Lec1

Goldberg machine: is a mechanical machine that searched for a pattern of dots or letters across Catalog entries stored on a roll of micro film

Information Retrieval: finding material of an unstructured nature that satisfies an information need from within large collections.

Data extraction: is a process that involves retrieval of data from various source in order to process it.

information extraction: is the automated retrieval of specific information related to a selected Topic bodies of text.

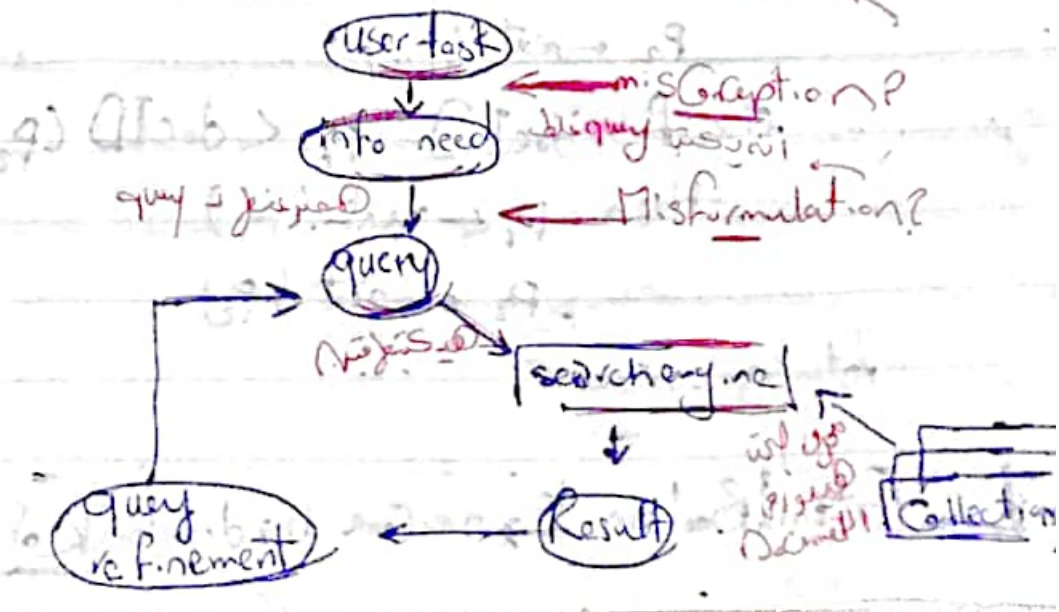
Data mining: is the process of analyzing of data to find patterns, discover trends and gain insight into how that data can be used.

web mining: is the process of using data mining techniques and algorithms.

web crawler: Called spider is a standalone bot that systematically scans the internet for indexing and searching for content, following internal links on web pages

web scraper: is a process of extracting specific data.
Unlike web crawling, a web scraper searches for specific information on specific websites or pages

The classic search Model



Lec 2

- information need: is the topic about which the user desires to know more
- query: is what the user conveys to the Computer in an attempt to Communicate the information need
- Relevance: if it is one that the user perceives as containing information of value with respect to their personal information need.

The effectiveness of an IR system

Precision: what fraction of the returned results are relevant to the information need?

Recall: what fraction of the relevant documents in the Collection were returned by the system?

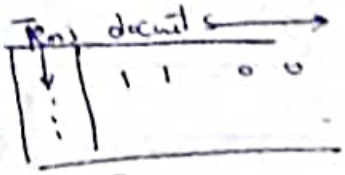
Grepping: the Unix Command allow useful possibilities for wildCard Pattern matching

very effective process

Shortfalls of Grepping

- To process Large document Collections quickly
- To allow more flexible matching operations
- To allow ranked retrieval

Term document incident matrix



query: O AND O

Boolean Retrieval Model: model for information retrieval in which we Can pose any query which is in the form of a Boolean expression of Terms

Lec 3

inverted index Construction

Document to be indexed → Friends Romans, Countrymen

Processing Document

Token stream → Tokenizer → Friends | Romans | Countrymen

Modified tokens → Linguistic modules → friend | roman | countryman

inverted index → indexer

friend → 2 → 4

roman → 1 → 2

countryman → 13 → 16

tokenizer → Normalize → Stemming → Stopword

Note Posting list are banded by the number of Terms lec 4

(Ex) Brutus and Caesar

Locate Brutus → Retrieve Position ①

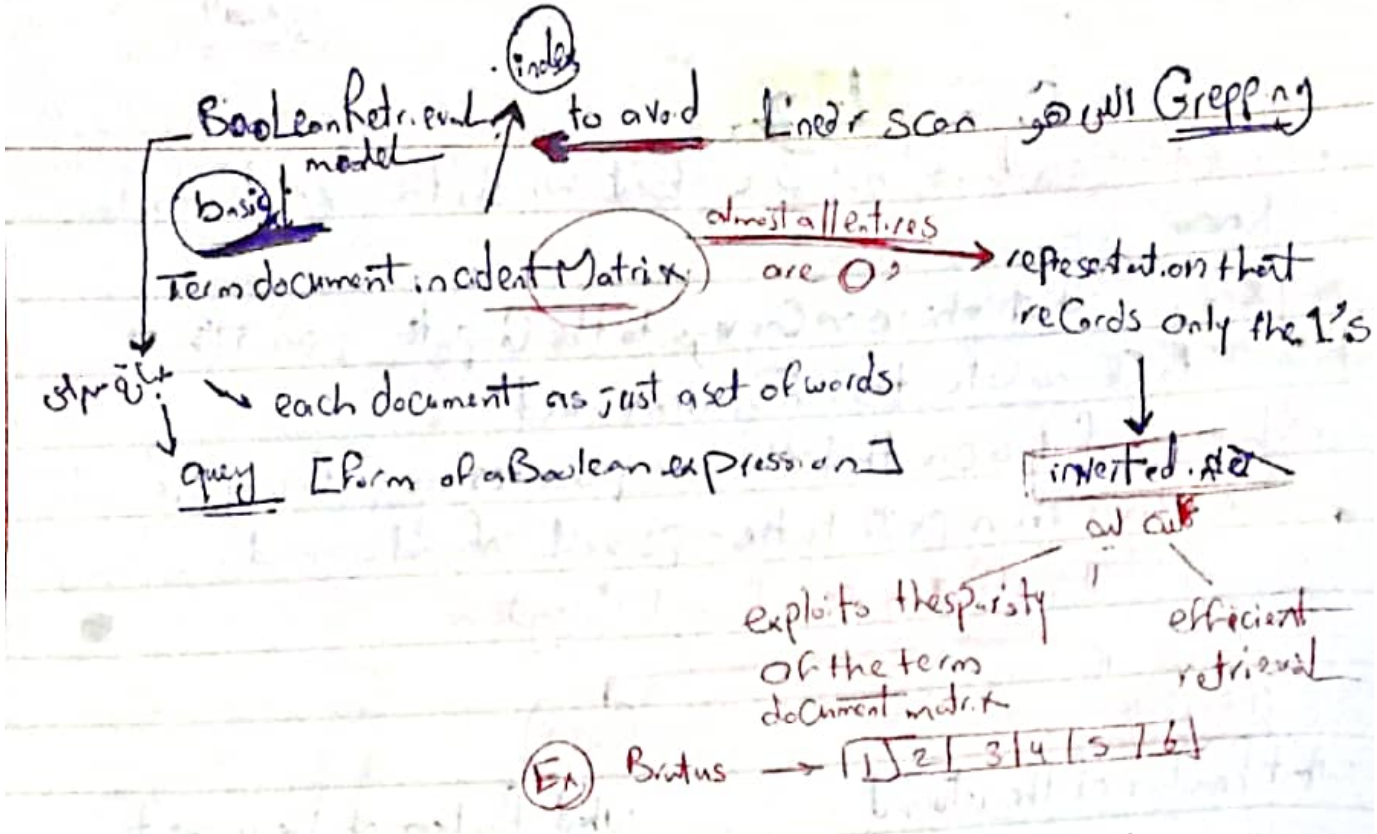
Locate Caesar → Retrieve Position ②

③ Merge Two Position

Processing Query

```
Intersect (P1, P2)
answer ← < >
while P1 ≠ NIL and P2 ≠ NIL
do if docID(P1) = docID(P2)
   then ADD (answer, docID(P1))
        P1 ← next(P1)
        P2 ← next(P2)
   elseif docID(P1) < docID(P2)
   then P1 ← next(P1)
        P2 ← next(P2)
return answer
```

↳ Extend Boolean to overcome the drawback of the Boolean (doesn't term weight



Forward index: is the list of documents, and which words appear in them

inverted index: is the list of words, and the document in which they appear

Westlaw → 13 within 3 words
↘ 15 in the Same Sentence

Query optimization
→ best order for query processing (start with Smallest

Lec3

Phrase Queries → want to able to answer Stanford University
(not match I want to University at Stanford)

First attempt
Biword indexes
→ each of biwords is a dictionary Term
Problem: Long phrase queries
Stanford University Palo Alto
→ Processed by breaking down
→ False Positive
- bigger dictionary
- not the Standard Solution

Combination Scheme

Solution 2
Positional indexes
= Store for each term the positions

```
Positional intersect (P1, P2, k)
answer ← < >
while P1 ≠ NIL, P2 ≠ NIL
do if docID(P1) = docID(P2)
   then L ← < >
        PP1 ← positions(P1)
        PP2 ← positions(P2)
   while PP1 ≠ NIL
   do while PP2 ≠ NIL
      do if |pos(PP1) - pos(PP2)| ≤ k
         then ADD(L, pos(PP2))
         elseif pos(PP2) > pos(PP1)
              then Break
      PP2 ← next(PP2)
   while L ≠ [ ] and |L[0] - pos(PP1)| > k
   do delete(L[0])
   Foreach pos ∈ L
   do ADD(answer, <docID(P1), pos(PP1), pos>)
   PP1 ← next(PP1)
   P1 ← next(P1)
   P2 ← next(P2)
else if docID(P1) < docID(P2)
     then P1 ← next(P1)
     else P2 ← next(P2)
return answer
```

عملي Lec 6

queries have all been Boolean

[→ Document match or not

query ودي مشكلة لان السيرش ماشيء ب query → problem → Feast or famine

(Ex) query 1: Stanford user dlink 650 → 200,000 hits

query 2: Stanford user dlink 650 no card found → 0 hit

Ranked retrieval ← حل

→ system returns an ordering over the top document in the collection with respect to query

→ Free text query

* Feast or famine not a problem in ranked retrieval?

because produces a ranked result

* (in the basic) score لكل Ranking عشان نعمل

document , query ال score لازم طريقة نحسب بيها

start with one term

document مش موجود في → score = 0

→ most frequent → higher the score

The Jaccard coefficient

→ measure of overlap of two sets

→ $jaccard(A,B) = \frac{|A \cap B|}{|A \cup B|}$

issues → doesn't consider term frequency

rare term are more information than frequent term

normalize = $\frac{A \cap B}{\sqrt{|A \cup B|}}$

Bag of words model

vector representation → problem → doesn't consider the ordering of words

(Ex) john is quicker than Mary = Mary is quicker than john

→ the same vector

able to distinguish the 2 document ← Positional index → Note

Term Frequency (tf) → the number of times that t occurs in d

Relevance doesn't increase with term TF

Use in Score

$w_{t,d} = 1 + \log tf_{t,d}$

Score = $\sum (1 + \log_{10} tf)$

Lec 7

df_t , is the document frequent of t : the number of documents that Contain t

no effect on Ranking one term queries ← $idf_t = \log\left(\frac{N}{df_t}\right)$

to dampen the effect of idf

- Collection Frequency of t is the number of occurence of t in the Collection, Counting multiple occurrence

$$w_{t,d} = (1 + \log tf_{t,d}) \cdot \log_{10}\left(\frac{N}{df}\right)$$

increase — # occurences within a document

rarity of term in the Collection

$score(q,d) = \sum tf \cdot idf$

Queries as vector → represent query as vector in the space

Euclidean slil distance بين نقط

bad idea

Because Euclidean distance is Large for vectors of different lengths

→ Use angel

angel between 2 doc is 0 Corresponding to maximal Similarity

Rank decrease

Cosine → decreasing function For interval [0, 180]

Rank increase

kind of inverse of angel

$Cos(q,d) = \frac{q \cdot d}{|q||d|} = \frac{\sum q \cdot d}{\sqrt{\sum q_i^2}\sqrt{\sum d_i^2}}$ similarity

q_t is the tf.idf weigh of Term t in the query

Crawler Need ← Key

Lec8

- Term Frequency ($1+\text{Log tf}$)
- Document Frequency ($\text{Log } \frac{N}{df}$)
- Normalization

① tf → ② $1+\text{Log}(\text{tf})$ → ③ df → ④ $\log(\frac{N}{df})$ → ⑤ tf·idf → ⑥ $(\omega)^2$, ± 1, Sum → ⑦ normalize $= \frac{w}{\sum w^2}$

- Computing Cosine Scores

```
CosineSCORE(q)
Float Scores[N] = 0
Float Length[N]
for each query term t
do calculate w_{t,q} and Fetch Postings list for t
    for each pair (d, tf_{t,d}) in Postings List
    do Scores[d] += w_{t,d} · x w_{t,q}
Read the array Length
for each d
do Scores[d] = Scores[d] / Length[d]
return Top k Components of Scores[]
```

Lec9

- Evaluation of a result (Measuring relevance)

if we have ① a benchmark document Collection
② a benchmark set of queries
③ assessor judgments of whether documents are relevant to queries

Then we can use Precision, ReCall, F-measure

- Evaluation of ranked results → the system can return any number of results
  → By taking various numbers of the top returned documents (levels of recall), the evaluator can produce a precision-recall curve

Measure for a search engine ① How fast does it index
② How fast does it search
③ UI ④ Free ⑤ Expressiveness of query language ⑥ user happiness

How do you tell: Users are happy?
- ① Search returns products relevant to users
- ② search results get clicked a lot
- ③ users buy after using the search engine
- ④ Repeat visitors / buyers

	Relevant	Non-relevant
Retrieved	TP (True positive)	FP
Not Retrieved	FN	TN

precision: fraction of retrieved docs that are relevant

Recall: fraction of relevant docs that are retrieved

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{FN + TP}$$

$$\text{F-measure} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

MAP is Average precision across multiple queries / rankings

Average precision

Recall 1/6 2/6 3/6 4/6 5/6 6/6

precision 1/1 2/2 3/3 0 0 0

average precision =

Map [mean average precision]

(Rank1 + Rank2) / 2

- if a relevant document never gets retrieved, we assume the precision corresponding to that relevant doc to be zero
- Map is macro-averaging: each query counts equally

lec 10

Basic crawler operation

- Begin with known seed URLs
- Fetch and parse them
  - → Extract URLs they point to
  - → place the extracted URLs on a queue
- Fetch each URL on the queue and repeat

Complications

Malicious Pages
- spam pages
- spider traps

non-malicious pages pose challenges
- bandwidth to remote servers vary
- web masters stipulations
- site mirrors and duplicate pages

spider trap

it is a set of web pages that may intentionally or unintentionally be used to cause a web crawler or search bot to make an infinite number of requests or cause a poorly constructed crawler to crash.

What any crawler must do

Be Robust

Be Polite
- Respect implicit and explicit politeness considerations
- no specification

Robots