Information Retrieval and Extraction

Course Logistics

- **3 major parts:**
    - 4 sessions: introduction from multiple aspects

    - 9? 11 : IR fundamentals:
        - models
            - scoring functions
            - crawling, evaluation, etc.
        NLP required specifically for IR
        ML for IR

    - : IE fundamentals
        - majorly: NER

    - : Information Access / Applications of IR
        - moving specifics: social media, sentiments,
        - computational advertising
        - sentiment analysis.

- <u>Tutorials</u>: Thu/Tue evening

- <u>Grading</u>:

Quizzes / In-class:	10%
Assignments (best 3/4)	15%
Project	60%

- Project:
    - mini   20 }
    - major   40 }

Term Paper	15%

(Final Exam, can be done anytime)

___ ✗ ___ ✗ ___

- **Recommended Books**
    - Stanford IR book
    - Search Engines in Practice

Project Details
- — <u>MINI</u>
    - Individual
    - 4 weeks, starting today
    - Deliverables:
        1.
        2.

    Lang:
        Py / C++ / Java

DEADLINES
    1. 24th August: offline
    2. 7th Sept: online + offline

> Design and develop a scalable and efficient search engine on wikipedia.

REQUIREMENTS
- Query $\xrightarrow{1s}$ Result
- Support "Field Queries"
- Total index size: < 1/4th the size of the doc repository
- build your own indexing scheme

EVAL

online {
- Search time
- search efficiency

offline {
- Indexing time
- Indexing size

1. Field Query: Searching within a subset.

MAJOR

- Team of 4 constrained choice
- 10 weeks.

Advanced topics
Scope well defined (by us)

- **5 touchpoints**
  report to mentor every 2 weeks

- **3 Evaluations**
  first deliverable: scope doc
  (26th Sept)

  MVP deliverable: full system, V1
  (25th Oct)

  Complete System: Demo, presentation report
  (14th Nov)   code, etc.

Basic Overview

- Searching is important
- Computational advertising

R E C A P

- **Information Retrieval**: Science of finding documents of unstructured information
- Economic viability of IR
- Standard Search Engines, and the new players (Fb, Amazon, etc)

———————— X ————————

History:

- **1993:**
  - new — WWW
  - no search engines
  - new URLs added manually to CERN "what's new"

    ┌───┐
    │ [Aside] — **1990:** • Archie. (Archive — v). First " search engine"   │
    │   • U.Minnisotta: ⟶ Veronica   ⟶ Jughead.   │
    └───┘

  - First .robo search program www Automated (developed at MIT)
  - Nov. 1993. 2nd websearch engine. (Aliweb)

- **1994:**
  - First proper webcrawler (also looks at content)
  - Another: Lycos, in CMU
  - Many others.

- **1995:**
  - Two prominent technologies:
    - crawling
    - indexing

1. CRAWLING:

- **problems:** 1a) Need diversity in seed URLs
  b) Need diversity in crawling strategy.

  2. Pages have a lifecycle.
     - TOI, Twitter change every minute
     - Something like IIIT changes occasionally.
     - Some websites never change.

     → **strat:** estimate lifecycles.
     1. After 1 crawl, crawl again.
        - if page has changed, increase freq. for it.
        - if page has not changed, decrease freq. for it.

2. INDEXING

- helps find URL and content.

- Many classes of indexing (schemes) exist.

Improvements to be made: coverage, scale

- Yahoo starts from Stanford in 1994.
  - Introduces the idea of "browse", searching enabled within a "topic directory".

- the web started growing faster than human editors could keep up
  - introduction of classification and clustering programs

- **1998: Google → PageRank**

  - Even if you know content of each page, that is not enough.
  - Value assigned by reputation of a given page.
  - Need to rank pages for query by reputation.

  - think of the web as a massive graph, where each URL has
    inlinks, and
    outlinks.
  - **Rank: the more inlinks there are, recursively.**

- **domain specific search engines:** let you incorporate more domain knowledge than generic search can do.

IR Cycle

Source Selection

Can be changed.

Documents containing your information. Could be WWW or enterprise DB

Query Formulation

Queries are processed and refined before being sent. Can use your context to improve the query. Like location, or in the direction you are driving.

Processing stuff like synonyms, NER, NLP for IR (chunk, POS, etc)

Search

Tries to prioritize which results are important. Ranking, for example.

Selection

Examination

Examining behaviour after first search to modify subsequent behaviour

Delivery.

- Tries to improve the search experience for the user.

- All of this happens while the user is typing.

- About:
  - quality
  - speed
  - variety of documents

Central problem with search

- As a searcher, I have information need in my head, expressed as a few terms. } Diff b/w concepts query terms

- As a document writer, I also have a concept in my head expressed as a few document terms. } Another diff.
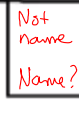
Both are suboptimal

- Job of search engine: Match the (suboptimal) Document Terms and Query Terms

Map the meaning, not the words
hence, search is complex.

Cross-Language IR system by IRE

- Auto identification of lang (when not unicode) hard
  - language ≠ script

Get synonyms, spellcheck, etc.

Query Parser → **Language Analyzer** → **Query Expansion** → **Query Translation** / **Query Transliteration**

Language Analyzer: *Does POS,NER, MwE, etc.*

Query Expansion: *Not name* / *Name?*

User Interface

Machine Translation ← **Summarization** ← **Document Ranking** ← **Document Retrieval** ← **Post-translation Query Expansion / Refinement**

QE, but after translating.

Clustering

Information Extraction

Index Models

Language Focused Crawler → **Language / Encoding Identifier** → **Encoding Converter** → **Documents** → **Indexer**

Abstract IR arch

Query

Docs

online | offline

Representation Function → Query representation

Representation Function → Document representation

Comparison function ← Index

Hits

- **How do we approach retrieving data**
  1. Looking at the content
  2. Looking at the meta data

20.3.20

Labeling by processing metadata ⎱ But this is not a replacement
  • No IR without content.
  • But metadata helps.

Document-query similarities

Much of IR depends on one idea: • Documents are similar to each
                                    other if they are relevant to
                                    the same query.
                                  • Likewise

• How is similarity measured?

  • {query vocab} ⊂ {doc. vocab}
  • prob model based on term occurences

  • Bag of words.

  • How to compare search engines? hard.
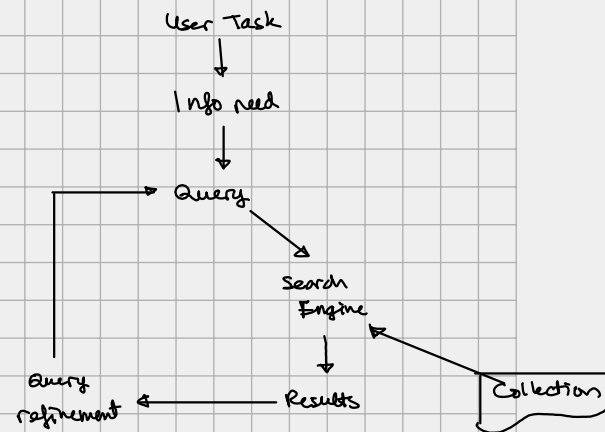   a given query will determine quality.

• Getting unstructured data hard.

Text Indexing

○ **Classic Search Model**

  - getting query from info need is
    search engine's job.

  - getting collection also hard but
    lets take as black box for now.

User Task
  ↓
Info need
  ↓
Query ──→ Search
  ↑   Engine
Query   ↓
refinement ← Results ← Collection

○ **Query on Unstructured Data**

  - boolean query for AND + NOT
  - could grep
      - slow
      - NOT is non-trivial
      - no ranking
      - no advanced search: like proximity

  ⇝ BUILD INDEXES _____

Reqs:
1. Indexing time smol
2. Indexing size ''
3. Query processing fast

1. **Term - Document Incidence Matrix:**

    - rows: vocabulary
    - cols: documents

Can bitwise AND/OR

Coming to bigger Collections

- say: N = 1 mill. documents, ~1000 words each
- ~6 GB data in the documents
- say: M = 500 K distinct terms.

> Standard TDI Matrix! 500 GB.
> but matrix is sparse (~ 1:499)

⇒ **Inverted Index**

- do not use fixed-size arrays

- Constructing
  - take documents

  tokenizer
  ↓

  Token stream
  |
  (ling. modules (stemming, etc)
  ↓

  ?? ?.

... Linguistics

... Sort
  - Sort by terms, then docID:

Search

| Map Reduce |
| Hadoop/ Spark |
| Saltom |