Information Retrieval & Web Search
Grid List

Ca' Foscari University of Venice
Department of Environmental Sciences, Informatics and Statistics

Academic Year 2021 - 222

Student Zuliani Riccardo 875532

Contents

Chapter 1

Introduction

- Definition of Information Retrieval
- Used by Web Search Engine, it has three main actors
  - User's perspective
  - Search Engine's perspective
  - Advertiser's perspective
- What makes webs search engine difficult
- Four main challenges:
  - Volume
  - Velocity
  - Variety
  - Varacity
- Web Search engine
  - Webcrawler
  - Indexing
  - Query Processing

Chapter 2

Basic Information Retrieval

- Assumption
  - Collection
  - Goal
- Measures
  - Precision
  - Recall
- **Term Document Incident Matrix**
  - Definition
  - Incident Matrix
  - Incident Vector
  - Disadvantage of Incident matrix
- **Inverted Index**
  - Definition
  - Construction
    1. Document gathering
    2. Tokenization
    3. Normalization
    4. Stemming
    5. Stop Words
    6. Inverted Index
  - Indexer Step:
    1. Token Sequence
    2. Sort
    3. Merge
- **Boolean Retrieval Model**
  - Definition

  - Why it works → AND, OR, NOT
  - Why it fails → AND, OR, NOT
  - Strengths
  - Weaknesses

- **Query optimization**
- **Phrase queries and positional indexes**
  - Problem of queries containing multiple words
  - Biwords indexes
  - Positional Indexes
  - Combination = Biwords + Positional Indexes
- **Fasting posting merge**
  - Skip pointers
- Structured VS Unstructured Data

Chapter 3

Document Ingestion

- Parsing Document
- Token and Tokenization
- Stop Words
- Normalization
  - Type
  - Terms
- Case Folding
- Thesauri and Soundex
- Lemmanization
- Stemming

Chapter 4

Dictionary Structure

- Native Dictionary
- How to store Dictionary
  - Hash Tables
    - Lookup in O(1)
    - No Prefix Search, rehashing, difficult to find mirror variants
  - Binary Trees
    - K keys = k+1 children
    - Solves prefix search
    - Search grows up to O(log M)
- **Wild Queries**
  - Definition
  - B-Tree
  - Permutation index
  - n-Gram index
- **Spelling Correction**
  - Definition
  - Isolated word
    - Assumption: correct lexicon
    - Definition
    - How to measure distances among words?
      - Edit distance: hamming distance
      - Weight edit distance
      - n-Gram overlap: jaccard coefficient
  - Context sensitive

Chapter 5

Index Construction

- **Hardware Basics**
  - Access to data
  - Disk Seeks
  - Transfer Speed
- **Sorting process**
  - Document are parsed to extract terms and saved with document id
  - Inverted file is sorted by terms and then by docID if the sorting algorithm is not stable
  - *Problem:* in memory index construction does not scale
    - Not big stuff in corpus, sorting by stable algorithm is possible by virtual memory
    - Sorting requires page fault and swapping
  - *Questions:*
    - How we can construct index for large corpus of data
    - Can we use the same index construction algo for larger collections, but by using disk insted of memory
- **BSBI**
  - Algo steps:
    - Accumulate postings for each block
    - Sort in memory block
    - Write to disk
    - Merge blocks into one and then write back
  - Binary merge? → rewrite increasigly larger blocks to disk: expensive
  - How we merge sorted runs?
  - Multi-way merge: read from all blocks simultaneosly
  - Need a dictionary to implement (term → termid) mapping
  - Work with term docid postings instead of termid docid postings

  - Intermediate file becomes very large

- **SPIMI**
  - Generate Separated dictionaries for each blocks (do not need the mapping across blocks)
  - Accumulate postings in postings list without sorting
  - With so we have a complete inverted index for each block, separate index can be merged into big index
  - Compression more efficient

- **Distribute indexing**
  - Distributed cluster computer to reduce data losses
  - Master Machine $\rightarrow$ Pool of Machines
  - Parses: read DAAT emits (term, docid) wrting into j-partitions
  - Inverter: for each j-partitions, they collect (term, docid), sort, write in posting
  - work divided in chucks, failure case reassign to individual workers

- **Dynamic Indexing**
  - Assumption: collections static, documents deleted and modified, dictionary and posing modified too
  - Periodically reconstruct the index from scratch
  - Maintain two indexes
    * Big $\rightarrow$ all terms, disk
    * Small $\rightarrow$ all new terms, memory
  - Search in both and merge results
  - Time in time we need to reindex in one main index with logarithmic merge

Chapter 6

Index Compression

- Compression allow us to
  - Inverted indexes
  - Posting lists
- Lossy and Lossness compression
- **Estimate Vocabulary Size**
  - How big is the term vocabulary, can we assume an upper bound?
  - Heap's Law
  - Zipf's Law
  - Power Law
- **Compression**
  - **Dictionary**
    - First cut
    - Dictionary as a string
    - Blocking
    - Front Coding
  - **Posting**
    - Posting file entry
    - Variable length coding
    - Gamma code
    - Delta code
    - Glomb-Rice
    - Interpolate Coding

Chapter 7

Scoring, Term Weighting and Vector Space model

- Query boolean, good for expert, bad for the majority of the user, boolean query results in too few or too many
- Rather a set of document satisfying a query expression in ranked retrieval, ordering over top docs in collection for query
- When a system provides a ranked result set, large result set are not an issue, we show the top k results
- How can we rank-order the documents in the collection with respect to the query?
  - Assign a score (0,1) to each doc, measure who well query doc match
    - If the query does not occur the score should be 0
    - The more frequent the query term in the document, the higher the score should be
- **How to assign a score**
  - Jaccard coefficient
    - It does not consider term frequency
    - Rare terms are more informative than frequent term
    - We need a more sophisticated way to normalize the length
  - Bag of words model → vector representation does not consider the ordering of words in a document, thus loose information
  - Term Frequency $TF_{t,d}$ → number of times that t occurs in document d
    - Relevance does not increase proportionally with term frequency
  - Log Frequency Weighting
    - mitigate previous problem, reason
  - IDF Weight
    - More emphaty for rare terms, less emphaty for common terms

    - * IDF has no effect on ranking "one term" queries, since all documents in which the term occurs have the same score
    - * IDF affects the ranking of documents for queries with at least two terms
  - **How do we weight document terms in the vectors?**
    - * terms that appear often in a document should get high weights
    - * terms that appear in many documents should get low weight
    - * How do we capture this mathematically speaking
  - **TF - IDF Weighting:** products of its weight and its IDS weight
- Binary, Count, Weight Matrix
- **Query as Vectors**
  - Do the same for queries: represent them as vectors in the space
  - Rank documents according to their proximity to the query
- We have two possible strategy:
  - **Euclidean Distance**, problem with the vector length
  - **Angle and Cosine**
    - * dot product
    - * length normalization
    - * cosine (query, document)

Chapter 8

Scoring and Result Assembly

- How to speed up the vector space ranking and analyse a complete search system
- TAAT: one query term at a time, features
- DAAT: compute the total score for each document before processing the next, features
- Efficient cosine ranking
  - Find the k documents in the collection nearest to the query: k largest query-doc cosines
    1. Computing the single cosine value efficiently
    2. Choosing the k largest cosine values efficiently
  - Can we do this without computing all the n cosines?
- **Safe Ranking:** guaranteed that the k docs returned are the k highest scoring documents
  - Max Heap to select the top K
  - Fast Scoring
  - Accumulator Array
  - Min Heap of K elements
  - Wanda Scoring
    - algo:
      1. Running threshold score (t-th)
      2. Prune away doc that are surely bellow the threshold
      3. Compute the exact score to the unpruned docs
    - index construction
    - upper bound
    - threashold
  - Block-max
    - wand sores the maximum "impact score" for each term/posting list

    - B-M index is an argumented inverted index structure to make wand faster
        - Stores the maximum "impact scores" for each block of compressed inverted list in uncompressed form
        - Split the inverted list into blocks to be decompressed separately
        - Table for each compressed block (info on that block)
    - Algo:
        1. Sort list from to to bottom via docid
        2. Find pivoit id
        3. Use the global maximum scores to check if the candidate pivoid is a real pivoit
        4. If is real:
        5. else:

- **Un-Safe Ranking:** the retrieved k doc may be accettable since ranking is only a proxy of user happiness. Its generic approch is to find a set A (in which it does not necessary contain the top k) of contenders. The primary computation bottleneck in scoring is the cosine computation of all candidates.
    - Wand made unsaefe
        - Safe aggreagre pruning: smaller k
        - unsafe aggregate pruning threashold
    - Index elimination: consider docs only containing one query term
        - Only consider High-IDF query terms
        - Documents that contain many query terms
    - Champion List
    - Static Query Score: we want the top-ranking docs to be relevant and authoritative
        - Relevant: modeled by cosine scores
        - Authoritative: query-independent propriety, so to model this we assign to each doc a quality score g(d)
        - Net Score
        - Top-k by net score
        - Champion list in g(d)-ordering
        - High and Low lists
        - Tired Indexes
        - Impact order postings
        - Cluster Pruning
        - Document clustering

- **Parametric and zone indexes**
    - Fields = (Author, title, date of pub, ...) = metadata of the document
    - Sometimes we wnat to search using these fields, thus we have to maintain a posting list for each field value

Chapter 9

Evaluation

- There are several ways to check if the users are satisfied of the search engine:
  - The search results get clicked a lot
  - Users buy some products after the search engine
  - Users and buyers repeat the search
  - Time user spend after clicking a link on a serp page before licking back to the serp results
- **Measuring relevance of search results**
  - Bench collection of documents
  - Bench suite of queries
  - Evaluation of either relevant or non relevant
- The evaluation should be based on the result satisfing the need and not on the content of required words
- **UnRanked Retrieval Evaluation**
  - Precision
  - Recall
  - F-Measure
- **Ranked Retrieval Evaluation**
  - Binary relevance
    - Precision(P@K)
    - Mean Average Precision (MAP)
    - Mean Reciprocal Rank (MAR)
  - Multiple Levels of relevance
    - Discounted Cumulative Gain
    - Normalized Discounted Cumulative Gain (NDCG)
- Using User Click
- Comparing to a basic ranking (Kendall Tan Distance)

Chapter 10

Relevance Feedback & Query Expansion

- Relevance to evaluate the quality of an IR system
- To evaluate we need three benchmarks
- The query result can be different from the real information needed
- Improving Recall
  - Increase relevance feedback: by telling the ir system which docs are relevant and non relevant
  - Query expansion: by adding synonyms and related terms
- **Relevance Feedback**
  - query, ret docs, user marks rel and non rel, se repres of info needed, se ret new res
  - Centroids
  - Rocchio's Algorithm
    - forula
    - maximize the similarity between rel docs, while minimizing the similarity with non rel docs
    - Cosine similarity
    - Rocchio smart algorithm
    - New queries move towards relevant documents and move away from non relevant documents
  - Problems:
    1. Expensive, because long modified queries are expensive to process
    2. Hard to understand way a particular cod was retreived
- **Pseudo-relevance Feedback**
- **Query Expansion**: method used to increase recall

- Publication or databases that collects (near)-synonyms, are called thesaurus
  - for each query t in the query we expand the query with words on the thesaurus, eg using wordnet
- Manual thesaurus
  - dictionary manually maintained, difficult to maintain and keeping updated
- Automatically derived thesaurus
  - Analysing the distribution of words in the documents using the similarity between words
    1. two words are similar if they co-occur with similar word
    2. two words are similar if they occur in a given grammatical relation with the same words
  - *co-occorrence thesaurus*: based on similarity in $C = A \cdot A^T$ where A is the document query matrix
  - *word- embedding*
    - dense vector for each word w
    - measure similarity as the vector dot scalar product
    - word2vec
    - word2vec for query expansion
- Query equivalence beased on query log minimg (query expansion in search engine)
  - query logs example

Chapter 11

Link Analysis

- Initially search engine compare content similarity of the query and the indexed pages
- Content similarity alone was no longer sufficient to express quality classification, since the number of pages grew and it was easy spammed
- Researches switch to **hyperlinks** (PageRank, HITS), which are used to find web communities
- **Network Proprieties**
  - (Barbasi - Albert) A network is scale free if its degree distribution follows a power law
  - A particular aspect is that the clustering coefficient C is very high (My friend are friend too)
  - Preferential attachment process (rich node gets richer), probability
  - Graph definition
  - SF network arise from the preferential attachment process, where it is a pattern of network growth, rich nodes get richer
  - Small world networks
- **Social Network Analysis**
  - Definition
  - Page = Social Actor
  - Hyperlink = Relationship
  - **Centrality**
    - Degree Centrality (Undirected, Directed)
    - Closness Centality
    - Betweeness Centrality (Undirected, Directed)
  - **Prestige**
    - Degree Prestige
    - Proximity Prestige

    - Rank Prestige

- **Co-citation and bibliographic coupling**
  - When a paper cite an other the relationship is establish
  - Co-citation
    - if paper i and j are both cited by paper k, then they may be related in some sense to one other. More paper they are cited by the stronger the relationship is. Similarity measure, number of cocite i and j
    - L is the similarity matrix
    - C_{ij} is the simialrity measure
  - Bibliographic coupling:
    - links paper that cite the same article. If i and j cite paper k, they may be related
    - the more paper cite both papers i and j the stronger their similarity is

- **PageRank**
  - Definition
  - More in links page i receives the more prestige it gets
  - The importance of page i is the sum of pagerank of all pages that points to i
  - p Columns vector of pagerank values
  - A Adjacent matrix
  - $P = A^T P$
  - Charatheristic equation of the eigensystem
  - $\lambda P = A^T P$ we use the power iteration to find p
  - Derive a Markov chain, to models the web surfing by a stochastic process
  - Propriety
  - Probability that the system is in state j after 1 step
  - Stationary probability distribution
  - A is not stochastic, not irreducible and not aperiodic
  - Proof
  - Teleportation
  - Advantage: fighting spam, global measure and it is query independent
  - Disadvantage: Query independent

- **HITS**
  - It is query independent
  - Authority: a good one is pointed by many god nodes
  - Hub: a good one points to many good authorities

- Scores
- Power iteration
- Strenght: ability to rank pages according to the query topic
- Weakness: easy spammed, topic drift, inefficient at query time

‘

Chapter 12

Web Crawling

- Process of locating, fetching and storing the pages available in the web
- Web crawlers exploit the hyperlinks structure of the web
- Basic crawler operation
  - Begin with known seed url
  - Fetch and parse them
    - Extract URLs they point to
    - Place the expected URLs on a queue
  - Fetch each URL on the queue and repeat
- Queue management is particular, in fact we could insert lot of spam pages or spider traps
- Consider variance of latency and stopping
- Crawler must be:
  - Robust
  - Explicit polite
  - Implicit Polite
- **What any crawler should do:**
  - Capable of distributed operation
  - Be scalable
  - Performance efficiency
  - Fetch page of "higher quality" first
  - Continuous operation
  - Extensible
- **Processing steps in crawling**
  1. Pick a url from the frontier

2. Fetch the document at the url
3. Parse the url, extracting the links from the other docs
4. Check if url has contents already seen, if not add to indexes
5. For each extracted url ensure it passes certain url filter tests and check if ti is already in the frontier

- **Basic Crawler Architecture:**
  - DNS
    - Problem asking sequential requests at a server
    - Solved by improving dns catching or sending a dns batch of requests
  - Parsing URL normalization: Solve problem of assign same url to different crawler
  - Content: if a fetched page is already in the index, do not further process it
  - Filters: verify rules in robots files, once a robots.txt file is fetched from a site, need not fetch it repeatedly. (This file specifies access restrictions)
  - Duplicate URL elimination: test if an extracted + filtered url has already been passed to the frontier
- **Distribute web crawling and fault tollerance**
  - Hash-based partitioning: stupid way since near site will be assign in 2 different locations
  - Site-based partitioning: arbitrary decide how to partition the space
  - *Fault tollerance*
    - Firewall Mode
    - Crossover Mode
    - Exchange Mode
- URL Frontier's considerations:
  - Freshness: crawl some pages more often than others
  - Politeness: do not hit a web server to often
- **URL Partition:** two separate queue for download of usrls
  - Discovery queue
  - Refreshing queue
- The web is dynamic
- Revisit policy of Web pages
  - General problem:
  - Freshness (binary) of page e_i
  - Age (real positive of page e_i

- **Repository Refreshing:**
  - Average freshness as two distinct policies:
    - Fresh page as high
    - age of pages as low
  - When Update:
    - Unifrom (better)
    - Proportional
  - Queue theory and quality of service

Chapter 13

Web Mining

Chapter 14

Learning to Rank

- Can we use ML to rank the documents displayed in the search result?
  - Why didn't it happen earlier?
  - Why wasn't much needed?
  - Why ML is needed now?
- Using classification for ad hoc IR
  - Training corpus: feature vector, shortest text span
  - Goal
  - Linear score function
  - Can we generalize these two classifier functions over more that two features?
- **BM25:** probabilistic model for scoring which use term independence assumption to approximate the document probability of being relevant
- **BM25F (multi-Fields)**
- How we find the best parameter? LTR framework
- LTR applied to BM25F
- Given a query q and a set of documents
  - Learn model h that allow to score and rank the documents D according to relevance
  - result = sort(h((d1),h(d2),...) h is the bm25f with proper parameter θ
- How we apply the gradient decent?
  - Gradient of sort wrt θ
  - Sort is not continuous and derivable function
  - How can we bypass this problem?
- *Line search*

  - Process
  - Guess the labels of the training set, a regressor where the objective is to optimize MSE rather than NDCG
  - Change the strategy considering the pairwise approach

- **RankNet**
- LRT applied to BM25F revisited
- LRT approches
- **Decision Tree**
  - Definition
  - Internal node and leaf node
  - How do we chose the best split
- **Gradient Boosted Regression Tree**
  - Combine weak learner sequentially
  - Evaluation with loss function
  - Lead to overfitting quickly
- **Lambda-MART**
  - Smooth approximation of the NDCG gradient, called λ-gradient
  - for each training instance, it estimates the benefit of incrementing or decreasing the currently predict score
  - λ_i single magic number for each url describing whether is well ranked and how much far is from it