DATA COLLECTION AND PREPARATION

Overview

- **Data:** measurement of something on a scale.
- **Metadata:** data about data (date, time, author, format, version, permissions, etc.)
- **Information:** data that has been processed, organized, and structured (has meaning)
- **Binary System:** 1MiB = 1024^2 bytes; 1MB = 1000^2 bytes
- **Information Life Cycle:**
  - **Occurrence:** discover, design, author
  - **Transmission:** networking, accessing, retrieving, transmitting
  - **Processing and Management:** collecting, validating, modifying, indexing, classifying, filtering, sorting, storing
  - **Usage:** monitoring, explaining, planning, forecasting, decision-making, educating, learning
- **Value in Data:** data is a source of value generation, providing evidence and content for the design of new products, new processes, and contribute to more efficient operations.
  - **Indirect value:** data provides influencing of supporting decisions (analysis in insurance, purchase decisions in retail)
  - **Direct value:** data provides value by feeding automated systems (search system, product recommendation system)
  - **How to increase data value?** Make data available, combine data, clean data, structure data, enrich data
- **Data-Intensive Applications**
  - **Problems:** amount of data available, complexity of data, speed at which it changes
  - **Tasks:** databases, caches, indexes, stream processing and batch processing
- **Data stages:**
  - **Raw:** data discovery, ingestion, understanding and metadata creation
  - **Refined:** data preparation for exploration, remove unwanted parts, reshape elements, establish relationships, assess data quality issues
  - **Production:** data integration into production process or products
- **Data Processing Patterns:**
  - **ETL:** Extract, Transform, Load
    - classic centralized IT driven operations
  - **ELT:** Extract, Load, Transform
    - evolution over ETL
    - systems can handle large volumes of data
    - column-oriented data structures for operations per field or property
    - allows division for responsibilities of data engineers and data analysts
  - **OSEMN:** Obtain, Scrub, Explore, Model, Interpret
    - although presented as a series of steps, real-word processes are typically non-linear

Data Collection

- **Open Data:** the idea that data should be freely available to anyone, to use, modify, and republish for any purpose.
- **Data Selection:** things to consider: is the source trustable? Is the data regularly updated? Does the data include information about how and when it was acquired? Does it seem plausible?

- **Ingestion Interfaces:**
  - Relational database behind an application: PostgreSQL, SQLite or Oracle
  - Layer of abstraction on top of a system: REST API
  - Endpoint to a message queue system: RabbitMQ
  - Shared network filesystem, containing logs, CSV files, or other flat files.
- **Data Structures:**
  - JSON from REST API
  - Well-structured data from a relational database
  - Semi-structured log data
  - CVS (comma-separated values) datasets
  - PDF, or other proprietary format files
  - HTML, or other semi-structured files
- **Data Encoding:** The process of translating from the in-memory representation to a byte sequence is called encoding (also known as serialization), and the inverse is called decoding (also parsing, deserialization).
  - **JSON, XML Serialization:** widely supported, human readable, used for data interchange between organizations/applications and as API outputs
  - **Binary Serialization:** more compact, faster to parse, works with images, used for within organization data interchange
- **Data quality:** missing data, outliers, inconsistent values, precision problems, duplicate vales, text encoding problems, mislabelled data, incomplete, outdated, etc.
- **Descriptive statistics:** common measures and techniques estimate the central tendency (mean, median, mode) and the dispersion (standard deviation, difference between max and min)

Data Preparation

- **Data Preparation:** data cleaning, data transformation, synthesis of data, data integration, data reduction or selection
- **Data Transformation:**
  - normalization of values to a comparable scale
  - scaling values to the same range
  - non-linear transformation to deal with skewed distributions
  - discretization/binning, which transforms numeric continuous values into ordered categorical values (eg: one category for $0<x<1$ values, another for $1<x<2$, etc.)
- **Synthesis of Data:** combine existing attributes to produce new additional attributes that are more convenient to analyse or use as input in follow-up phases
- **Data Integration:** combine data that originally exists in multiple sources.
- **Data Reduction or Selection:** if data is not relevant, is outdated, if data volume exceeds exiting capacity for processing, or if existing precision is excessive while lower precision is sufficient.
  - **Data Filtering:** used to remove data with unsuitable values, data that is irrelevant for the scope of the project, outdated data items, legal/dev reasons, etc
  - **Data Sampling:** non-deterministic process that takes a random subset of the data items of a requested size (must be representative of the complete collection)
  - **Data Aggregation:** may be used to reduce excessive detail in data
- **Visualization in Data Preparation:** can be applied before/during/after the application of computational methods to assess data quality and changes

DATA PROCESSING

Data Storage

- **Data Models:**
  - **Relational Model:** data in organized in relations (sql tables), which consist of an unordered collection of tuples (sql rows).
    - best one for support joins, N-1 and N-N relationships
    - needs a translation layer between tables and objects in object-oriented programming
    - NoSQL provides better scalability, specialized query operations and a more flexible and expressive data model.
  - **Document Model:** data is organized as singles instances (documents) and they are stored together with their parent records in document-oriented databases.
    - best one for one-to-many relationships (tree-like structures) and no relationships between records
    - schema flexibility
    - better performance (related data is stored together)
    - data model is closer to application's data structures.
  - **Property Graph Model:** vertices define entities and edges define relationships
    - best for many-to-many relationships
    - provides a consistent way of storing completely different types of data in a single datastore (graphs are not limited to homogeneous data)
- **Schema Flexibility:**
  - **no schema (schemaless):** arbitrary keys and values can be added to a document and, when reading, clients have no guarantees as to what fields to expect
  - Although schemaless databases do not impose a schema, code that reads the data usually assumes some kind of structure, thus there is an implicit schema, but not enforced by the database! Ideal for when the items in a collection don't have the same structure, or the structure of data is determined by external systems which may change over time.
  - **schema-on-read:** the structure is enforced when data is read, at code level
  - **schema-on-write:** the structure is enforced by the database (traditional relational approach)

Batch Processing

- **Interaction Types:**
  - **Online systems (services):** a service waits for requests or instructions from a client to arrive. Response time is the primary measure of performance.
  - **Offline systems (batch processing):** a batch processing system takes a large amount of input data, runs a job to process it, and produces some output data. Suitable for long jobs or asynchronous processes.
  - **Stream processing systems:** a service operates on inputs and produces outputs (rather than responses) as a result of an event happening (rather than periodically at scheduled times).
- **Batch Processing:** using a sequence of UNIX commands relies on sorting as a key step in preparing data for counting, while a custom program loads all data into memory.
- **Unix Philosophy:**
  1. Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new features.

2. Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.
3. Design and build software, even operating systems, to be tried early, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.
4. Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

- **Unix Uniform Interface:**
  - everything is a file
  - filesystem files, TCP connections, device drivers, (…) all share the same uniform interface, and thus can be plugged together, building a pipe
  - Unix tools run only on a single machine (limitation overcome by Hadoop)

Makefiles

- **Make:** software tool that controls the execution of commands to derive target files, based on the existence (or change) of other files (dependencies) and the execution of operations (recipes). It can generally be used in scenarios where the execution workflow can be encapsulated as a set of dependencies and execution rules. Makefiles support control structures and building in parallel.

Data Presentation

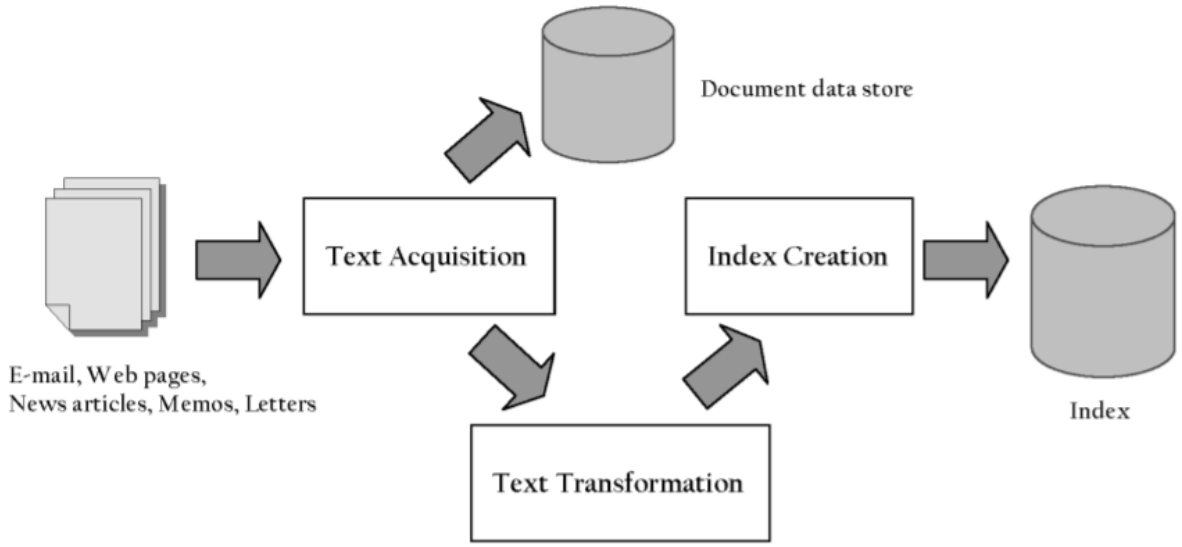
- **Data Visualization:** algorithmically drawn, easy to regenerate, and aesthetically simple visualizations
  - **Exploration:** is generally done at a higher level of granularity, earlier in the process, and the main goal is to understand what is in the data
  - **Explanation:** is appropriate when you already know the data and you want to support and present findings
- **Elements of Visualization:** charts, time-series, maps, interactive, words

IR BASICS

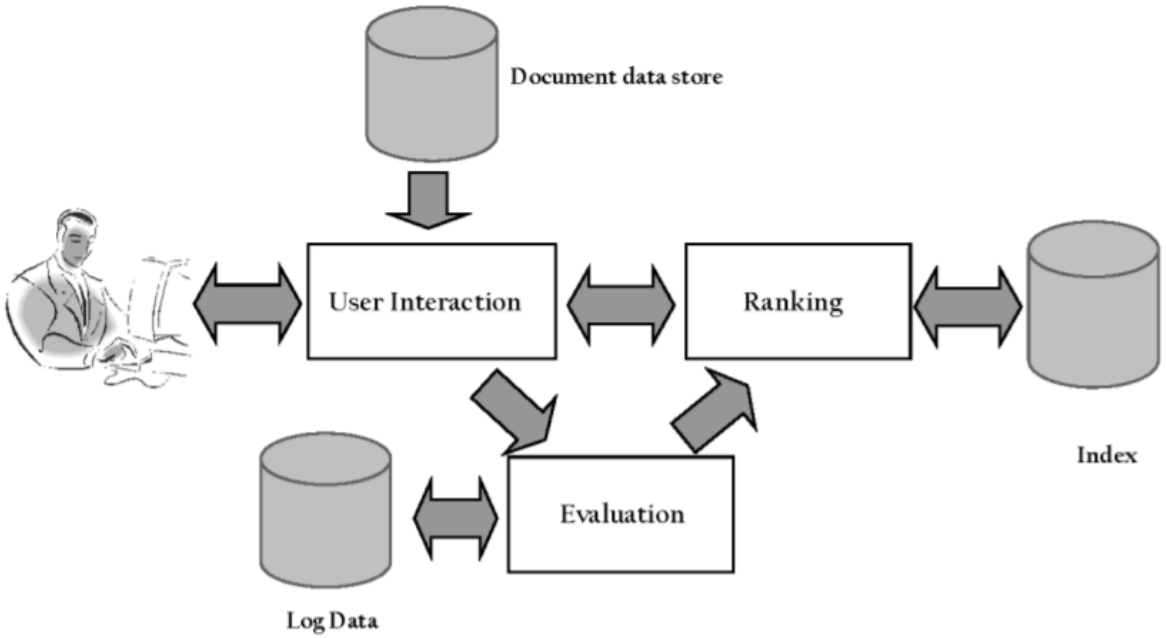
- **Information retrieval:** field concerned with the structure, analysis, organization, storage, searching, and retrieval of information.
- **Document:** significant text content with some structure; used in search engine queries (web pages, books, pdf, etc)
- **Records:** made up of well-defined fields; easy to compare fields with well-defined semantics to queries in order to find matches; text is more difficult (bank records with account numbers, balances, etc)
- **IR Tasks:** ad-hoc search, filtering, classification, question answering
- **IR Issues:**
  - **Relevance:** A relevant document contains the information that a person was looking for when they submitted a query to the search engine. Retrieval models define a view of relevance. Ranking algorithms used in search engines are based on retrieval models.
  - **Evaluation:** comparison of system output with user expectations (TREC collections for testing; recall and precision for effectiveness measurement)

- **Users and IN:** Keyword queries are often poor descriptions of actual information needs. Interaction and context are important for understanding user intent. Query refinement techniques such as query expansion, query suggestion, relevance feedback improve ranking.

- **Search Engine:**
  - **Tasks/Issues:**
    - **Performance**: Efficient search and indexing
    - **Dynamic Data:** Coverage and freshness of new data
    - **Scalability:** Growing with data and users
    - **Adaptability:** Tuning for applications
    - **Spam**
  - **Indexing Processing:**



  - **Query Process**



IR EVALUATION

- Evaluation depends on the task, collection, and information need type (**informational** if looking for multiple results, or **navigational** if looking for "one destiny")
- Evaluation is important to understand the use of a system by its users and make decisions on new designs and features to implement
- **Effectiveness:** measures the ability of a search system to find the right information.
- **Efficiency:** measures how quickly a search system provides an answer.
- To measure the effectiveness, we need a **document collection**, a **test suite of information needs**, expressible as queries, and a **set of relevance judgements**, labelled as relevant and non-relevant.
- Relevance is assessed relative to an information need, not a query
- **Cranfield Experiments:** Cyril Cleverdon introduced the notion of test reference collections, composed of documents, queries, and relevance judgements. Reference collections allows using the same set of documents and queries to evaluate different ranking systems

	relevant	not relevant
retrieved	true positives (tp)	false positives (fp)
not retrieved	false negatives (fn)	true negatives (tn)

- **Precision:** (relevant retrieved) / (all retrieved) or TP / (TP + FP)
- **Recall:** (relevant retrieved) / (all relevant) or TP / (TP + FN)
- **Accuracy:** (TP + TN) / (TP + FP + FN + TN)
- **F score:** weighted harmonic mean of precision (P) and recall (R), $F = (1 + \beta^2) \times \frac{P \times R}{\beta^2 \times P + R}$ possible to emphasize Precision (β < 1) or Recall (β > 1).
- **P@k:** In web search, high recall isn't required. What matters are high quality results on the first page. This leads to measuring precision at fixed low levels of retrieved result (P@5 means precision at 5)
- **Average Precision (AvP):** for a single information need, AvP is the average of the precision value obtained for the set of top k documents existing after each relevant document is retrieved.
- **Mean Average Precision (MAP):** given a set of queries, MAP is the mean over the AvP values.
- **User-Centred Evaluation:** observe user in controlled sessions; observe server-side query logs to measure performance; use client-side logging tools to study interaction patterns
- **Online Evaluation:** large-scale evaluation experiments can be made with A/B tests
- **Efficiency Metrics:** elapsed indexing time, indexing processor time, query throughput, query latency, indexing temporary space, index size

IR MODELS AND INDEXING

- **Incidence Matrix:** matrix element is 1 if given word is contained in a document, 0 otherwise
- **Boolean Model:** queries are represented in the form of a boolean expression of terms and the model views each document as a bag of words
- **Inverted Index:** dictionary of words and a list of pointers for each document they appear in
- **Index Construction:** choosing the document unit and the index granularity is important to choose the trade off between missing important content and keeping the results relevant.

- **Tokenization:** task of chopping a character sequence into semantically useful pieces, called tokens
- **Stop Words:** words considered of little value in helping select documents in the retrieval process (typically very frequent in all documents, eg: a an and are as for from has is it of that the to …)
- **Token Normalization:** canonicalizing tokens so that matches occur despite superficial differences, like accents, capitalization and stemming.

Term Weighting

- **Ranked Retrieval:** for when Boolean model isn't feasible due to large document collections, returning a ranked order of documents to a search query.
- **Parametric Indexes:** inverted indexes applied to specific parameters/fields, supporting parametric search (eg: all documents from author Z containing word Y)
- **Parametric Zones:** similar concept of parametric indexes, but applied to arbitrary free text
- **Ranked Boolean Retrieval:** zones and fields can be weighted differently to compute each document's relevance simply using a linear combination of zone scores, where each zone of the document contributes a Boolean value
- **Term Frequency (tf):** number of occurrences of a term in a document
- **Document Frequency (df):** number of documents that contain a given term
- **Inverse Document Frequency (idf):** N being the total number of documents. The rarer the term is in a collection, the higher its idf is.

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}.$$

- **tf-idf:** assigns a term t a weight in a document d that is
  - highest when t occurs many times within a small number of documents
  - lower when the term occurs fewer times in a document, or occurs in many documents
  - lowest when the term occurs in virtually all documents

Vector Space Model

- **Vector Space Mode:** each document is represented as a vector, with a component vector for each dictionary term. tf-idf weights are used as components. Thus, the set of documents in a collection may be viewed as a set of vectors in a vector space, in which there is one axis for each term.
- **Cosine Similarity:** the similarity between two documents is given by the cosine of the angle between the two vector representations of the documents

$$\mathrm{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|},$$

- **Queries as Vectors:** Queries can also be represented as vectors in a n-dimensional space, being n the number of terms in the query. Basically, queries are viewed as very short documents. The top ranked results for a given query are thus the documents whose vectors have the highest cosine similarity in comparison with the query vector.

Language Models

- **Document Model:** function that returns a probability of a word being drawn from some document
  - **Unigram Model:** discards all the context information and estimates the probability of each term independently. Main model used in IR.

$$P_{unigram}(t_1t_2t_3t_4) = P(t_1) \times P(t_2) \times P(t_3) \times P(t_4)$$

  - **Bigram Model:** estimates the probability of each term on the previous item. Used in speech recognition, spelling correction and machine translation

$$P_{bigram}(t_1t_2t_3t_4) = P(t_1) \times P(t_2|t_1) \times P(t_3|t_2) \times P(t_4|t_3)$$

- **Language Model Retrieval:**
  1. Infer a LM for each document
  2. Estimate the probability of generating the query according to each document model $P(q|M_{di})$
  3. Rank the documents according to these probabilities

WEB IR

- **Information Access on the Web**
  - **Full-text index search engines:** Users use keyword-based search supported by inverted indexes and ranking mechanisms (eg: Altavista, Excite, Infoseek)
  - **Directories / Taxonomies:** Users navigate through a hierarchical tree of category labels. Problems: mostly manual categorization (high cost and not scalable) and mismatch between editors' and users' idea of how to organize a given node. (eg: Yahoo!, Open Directory Project)
- **Web Search Challenges:** Decentralization of content publication, content is created massively and in diverse forms (languages, formats, quality, intent), uncertainty of the size of the web.
- **Web Characteristics:** the web can be modelled as a graph (bowtie shape) and pages point to, and are pointed by, other pages. Out-links usually include “anchor text” and the number of in-links is called “in-degree”.
- **Web Spam:** manipulation of content on the web with the purpose of manipulating search engine rankings (cloaking, link farms, link spam, click spam, etc)
  - **Cloaking:** web page is shown differently to the search engine and the user
- **Search Engine Economic Model:**
  - Earlier, banner advertisements were priced based on cost per mil (CPM) impressions
  - Alternatively, contracts can be set based on the cost per click (CPC)
  - Currently, mix of CPC and bidding between ad companies to place ads based on keywords
  - Additionally, targeting ads explore contextual or behavioural user data
- **User Characteristics:** Studies in search user behaviour show that users use between two or three keywords, search operators are rarely used, precision in the top results is highly valued, and lightweight result pages are preferred.
  - **Informational queries:** seek general information about a topic, typically from multiple web pages.
  - **Navigational queries:** seek the website or home page of a single specific entity
  - **Transactional queries:** intent of a transaction on the web, such as purchasing a product, downloading a file, or doing a reservation.
- **Signals for Web Ranking:** Query dependent and independent signals, document-based signals (content or structural), collection-based signals (links), user-based signals (clicks)

Web Crawling

- **Web Crawling:** process by which pages are gathered from the web
  - **must:** robustness in face of problems or traps, politeness to web hosts
  - **should:** execute in a distributed fashion, scalable, efficient use of resources, bias towards good quality pages, freshness depending on page change rate, and extensible to cope with innovations on the web.

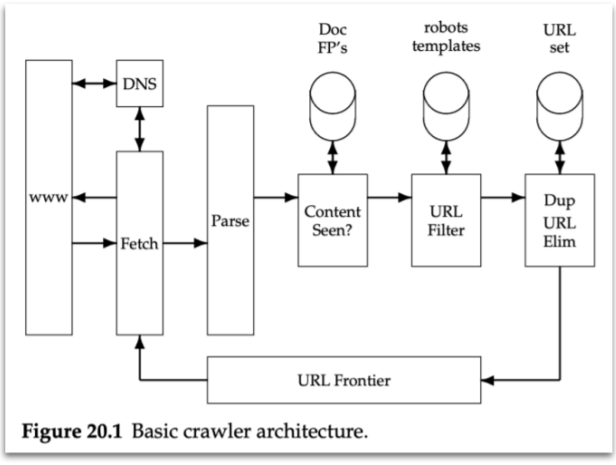
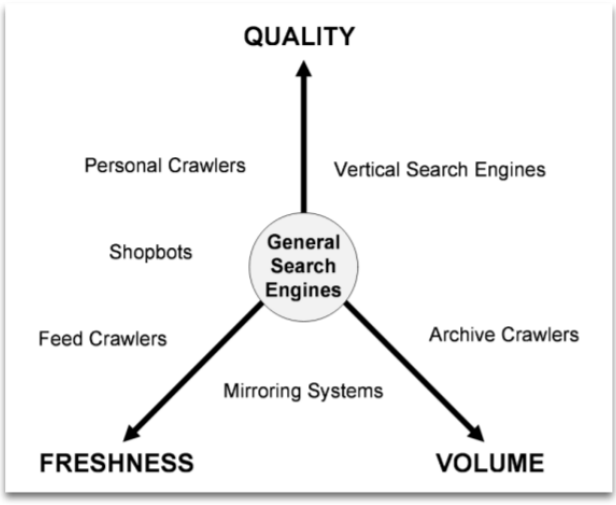


Figure 20.1 Basic crawler architecture.



- **Near-Duplicate Detection:** A large percentage of content on the web are near-duplicates. Crawlers need to decide if new pages are duplicates of existing content and if pages being revisited have changed since last visit (estimate change rate). Common solution: obtain shingles of k size from web pages; compare shingles from two pages to determine if they are near-duplicates. The more shingles in common, the more similar are the pages.

Link Analysis

- **Link-based Signals:** links are one of the distinctive features of a collection of web documents. It's assumed that the number of hyperlinks pointing to a page provides a measure of its popularity and quality. Link-based ranking algorithms assume that an hyperlink from page A to page B represents an endorsement of page B, by the creator of page A
  - **PageRank:** value between 0 and 1. query-independent value computed offline that only depends on the structure of the web graph. Uses a random surfer to visit out-links to move from page to page, executing "forever", saving the frequency of visited pages.
  - **HITS (Hyperlink Induced Topic Search):** query-dependent algorithm that computes two scores for each page: authority score and hub score.
    - pages with many links pointing to them are called **authorities**.
    - pages with many outgoing links are called **hubs**.
- **Anchor Text as Signal:** The collection of all anchor texts can be explored with standard IR techniques, and incorporated as an additional feature in an inverted index (can be exploited using Google Bombing)

QUERY PROCESSING

- **Document-at-a-time:** calculates complete scores for documents by processing all term lists, one document at a time. At the end all documents are sorted according to their score
- **Term-at-a-time:** accumulates scores for documents by processing term lists one at a time. When all terms are processed, the accumulators contain the final scores of all matching documents
- **Optimization Techniques:** focus on reading less data from the index and processing fewer docs
  - **Skip Pointers:** used to speed-up inverted index scans
  - **Conjunctive Processing:** only return documents that contain all query terms, starting with the rarest one. Efficient and effective with short queries, not ideal with long queries
  - **Early termination of query processing:** ignore high-frequency words in term-at-a-time and documents at the end pf quality ordered lists in document-at-a-time
  - **Order postings in inverted indexes:** order inverted lists by some quality metric
  - **Caching:** caching popular query results

Relevance Feedback and Query Expansion

- **Relevance Feedback:**
  - exact matches aren't the only way to obtain relevant results in search systems
  - vocabulary mismatch between the user and the collection and/or use of synonyms
  - relevance feedback consists in involving the user in the process to improve the final result set, by considering user feedback on about the initial set of results.
  - it may be difficult to formulate a good query when you don't know the collection, but it is easy to judge particular documents
  - seeing some documents may lead users to refine their understanding of the original information need.

- **Relevance Feedback Local Methods:** adjust a query relative to the documents that initially appear to match the query
  - **Rocchio Algorithm:**
    - classic algorithm for implementing relevance feedback by representing information in a vector space model
    - based on the concept of an optimal query vector, which maximizes the difference between the average vector representing the relevant documents, and the average vector representing the non-relevant documents
    - considering that only partial relevance information is available about the collection, the Rocchio algorithm defines the "modified query" as a weighted combination of the initial query and the difference vector between the centroid of the documents marked as relevant and the centroid of the documents marked as non-relevant
    - can improve both recall and precision, but is most useful for increasing recall
  - **Relevance Feedback Limitations:** misspelling, cross-language retrieval, vocabulary mismatch.
  - **Pseudo Relevance Feedback:** assumes that the top k ranked documents are relevant and apply relevance feedback under this assumption (automates the manual part, no extended user interaction)
  - **Implicit Relevance Feedback:** use indirect sources of evidence rather than explicit feedback on relevance (contains evidence of user judgements, eg: cickstreams)
- **Relevance Feedback Global Methods:** expand or reformulate the query terms independently of the query or the results
  - **Query Expansion:** user give additional input on query words or phrases to suggest additional terms. Users opt to use one of alternative query suggestions, generated by the use of synonyms, related words from a global thesaurus and user-specific personalized data

Search User Experience

- **Search User Experience:** Ranking is only a part of the information retrieval process, user interaction is central to the overall user experience and success of the user task.
  - Support natural language queries.
  - Query auto-complete and suggestions;
  - Results snippets, eg: query-dependent result snippets.
  - Clustering results.
  - Support site search.