Exam Study Guide, PRI 2023/24

Information Processing and Retrieval (PRI)

Master in Informatics Engineering and Computation (M.EIC)

FEUP, U.Porto

Completed by Sofia Merino Costa
up202300565

Completed by Sofia Merino Costa
up202300565

Introduction

This is an **incomplete** guide to help you on your preparation for the exam. It includes a set of recommendations about the topics, available materials, **example questions**, and references.

The exam is answered in Moodle and has a **maximum duration of 75 minutes**. It is a closed-book exam, but some reference materials are made available for students to use during the exam. The exam includes **multiple-choice questions** (with a penalty of 1/3 for wrong answers) **and short open questions**. It also includes questions concerning the group projects.

The “Introduction to Information Retrieval” book, from Manning et al. (2008), will be available during the exam in PDF format through Moodle.

Completed by Sofia Merino Costa
up202300565

Topics

Some topics for which there will be questions:

1. **Information processing:** concepts, techniques;
2. **Information retrieval:** information need, search task, collection, query, results list;
3. **Search systems:** concepts, indexing and retrieval;
4. Building **inverted indexes**;
5. **Vector model:** tf and idf calculations to compose the weight of a term in a document;
6. **Vector model:** calculate the score of a document for a query;
7. **Evaluation:** calculate recall and precision, draw P versus R curves (with average interpolated precision), calculate MAP;
8. **Web retrieval and link analysis:** PageRank, hubs and authorities;
9. **Query processing:** query processing techniques and relevance feedback strategies;
10. **Neural Information Retrieval:** dense vectors, embeddings, using neural models in the retrieval process.
11. **Learning to Rank:** using machine learning in the retrieval process.
12. **Entity-oriented search:** producing entity descriptions, entity ranking and entity linking;
13. **Search user interfaces:** user interaction with search systems, interaction elements.

Some detailed references

- BY refers to Modern Information Retrieval, by Baeza-Yates and Ribeiro-Neto
- Manning refers to Introduction to Information Retrieval, by Manning et al.
- Croft refers to Search Engines in Practice, by Croft et al.
- Balog refers to Entity-Oriented Search, by Balog.

Completed by Sofia Merino Costa
up202300565

1. Information processing (L2)

Data:

- is a measurement of something on a scale;
- a fact known by direct observation.

Metadata:

- is "data about data";
- not the content of data but data providing information about one or more aspects of the data, such as description (date, time, author), structure (format, version), administrative (permissions), legal, etc.

Information:

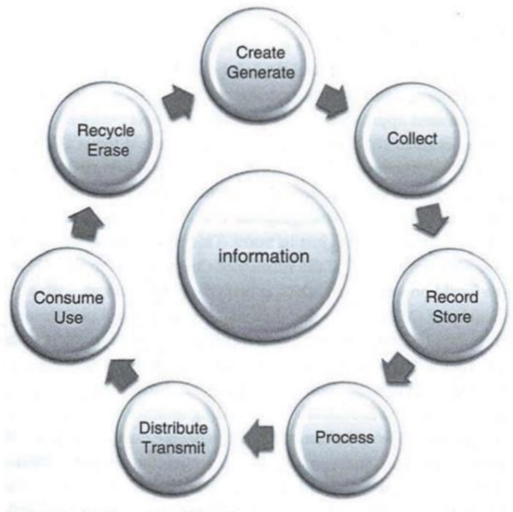
- is data with a context / meaning, thus enabling decision making;
- is data that has been processed, organized and structured.

Terminology: Decimal and Binary Systems

- The **binary** system uses power of 2 units.
- The **decimal** system uses power of 10 units.

Information Life Cycle:

- The life cycle of information typically includes the following phases:
    - Occurrence: discover, design, author, etc;
    - Transmission: networking, accessing, retrieving, transmitting, etc;
    - Processing and Management: collecting, validating, modifying, indexing, classifying, filtering, sorting, storing, etc;
    - Usage: monitoring, explaining, planning, forecasting, decision-making, educating, learning, etc;



1. A typical information life cycle

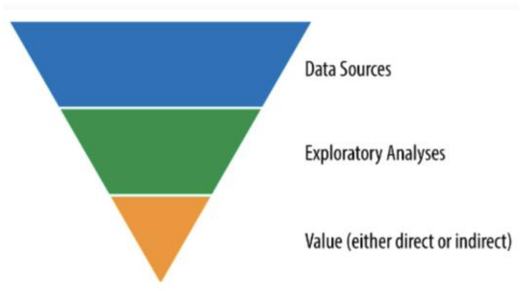
- **Creation/Generation**: This is the initial phase where information is generated or created. It may originate from various sources such as individuals, organizations, sensors, or automated systems.
- **Collect**: Following creation, this phase involves the gathering of information from various sources. Collection may include manual entry, data acquisition from sensors, or ingestion from external systems.
- **Record/Store**: In this phase, the collected information is stored or recorded for future reference. This can involve organizing the data, applying metadata, and storing it in appropriate data repositories.
- **Process**: After storage, the information may undergo processing. This phase involves manipulating, analysing, or transforming the data to derive insights, make decisions, or prepare it for further use.
- **Distribute/Transmit**: Once processed, information is shared or transmitted to relevant parties. This could involve distribution through various channels such as emails, reports, or collaborative platforms.
- **Consume/Use**: In this phase, users or systems access and utilize the information for specific purposes. It includes activities like analysis, decision-making, reporting, or any other form of utilization.
- **Recycle/Erase**: This phase involves managing the end of the information's lifecycle. Information that is no longer needed is either recycled, meaning it might be repurposed or reused, or erased, indicating its secure deletion or disposal.

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.
- In data-driven approaches, multidisciplinary teams experiment and explore large and diverse sources of data to "extract signal from the noise".
- Indirect value — data provides value by influencing of supporting decisions, e.g. risk analysis in insurance, purchase decisions in retail.
- Direct value — data provides value by feeding automated systems, e.g. search system, product recommendation system.

Data Value Funnel:

- A large number of data sources and exploratory analysis are required to produce a single valuable application of data.
- Minimize the time spent on non-relevant data by empowering business-experts (i.e. people who know about the business) to explore data.
- Additionally, make data analysis processes as efficient as possible, e.g. by implementing effective data processing workflows.



Increasing Data Value:

- Make data available, i.e. simply make previously inaccessible data, available.
- Combine data, i.e. create a single coherent whole from disperse data sources; e.g. collection of news from the main news outlets during a year.
- Clean data, i.e. eliminate problems such as incomplete values, duplicates; or create a subset according to specific criteria.

- Structure data, i.e. provide structure to unstructured data; e.g. derive a mentioned entities field from a textual field.
- Enrich data, i.e. complement existing data with data from other sources, including the computation necessary to do so.

Data-Intensive Applications:

- Many applications today are data-intensive, as opposed to computing-intensive. In this context, existing problems are:
  - the amount of data available;
  - the complexity of the data;
  - the speed at which it changes.
- Common building blocks in data-intensive application include:
  - Store data, for use or sharing (databases);
  - Send messages between systems (stream processing);
  - Periodically process large amount of data (batch processing);
  - Remember the result of an expensive operation (caches);
  - Enable search and filtering (indexes)

Data Stages:

- Data moves through three main stages:
  - **Raw** — focus is on data discovery; the primary goals are ingestion, understanding, and metadata creation; common questions include: what kinds of records are in the data? how are record fields encoded?
  - **Refined** — focus is on data preparation for further exploration; tasks include removing unwanted parts, reshaping poorly formatted elements; establishing relationships between datasets; assessing data quality issues.
  - **Production** — focus is on integrating the data into production processes or products.

ETL Pattern:

- The **ETL framework (extract-transform-load)** was coined in the 1970s and popularized in the context of data warehousing.

    - **Extract** involves extracting data from the source system.
    - **Transform**, a series of operations or transformations are applied to the extracted data.
    - **Load** involves publishing data to the target system, either simple flat files or other infrastructures.
- ETL is usually associated with classic centralized IT driven operations.

ELT and EtLT Frameworks:

- **ELT (extract-load-transform)** is a recent evolution over the ETL framework. Load-transform, in contrast with transform-load, is a pattern more well-suited to the
- division of responsibilities in multidisciplinary teams.
- Increasingly common access to data storage infrastructures capable of handling large volumes of data has lead to a more flexible pattern.
- Column-oriented data structures are particularly well-suited to typical data processing tasks, i.e. organizing operations per field or property.
- The **sub-pattern EtLT** introduces a transformation step before the loading, typically associated with data cleaning tasks.

OSEMN Framework:

- In the context of Data Science, the **OSEMN** (pronounced "awesome") was coined.
    - **Obtain**, gathering data.
    - **Scrub**, clear, arrange, prepare data.
    - **Explore**, observe, experiment, visualize.
    - **Model**, create a statistical model of the data.
    - **Interpret**, drawn conclusions, evaluating and communicating results.

Data Analytics Process in Organizations:

- Data analytics processes can either be centralized, typically in an IT department, or decentralized, in specialized teams.
- **Benefits of centrally** controlling data processing:
    - Controlled data governance;
    - Efficiency gains due to reuse of practices, methods, and expertise.

- **Drawbacks of centralization**:
    - Frequent bottlenecks due to the time taken to dependency upon the IT department
- **Challenge**: expanding the range of users who have access to raw data and provide them with the necessary training and skills.

Data Engineers:

- Data engineers have emerged as an autonomous key role in this context.
- Design, implement and maintain data processing pipelines.
- Work closely with data scientists and analysts to understand what will be done with the data.
- Wide range of technical skills:
    - SQL and Data Warehousing
    - Programming - Python, Java, Go (common in this context)
    - Distributed Computing
    - Cloud Infrastructures
    - System Administration

Questions:

- **1. Distinguish between data, metadata, and information.**

- **2. Identify and describe the phases of a typical information lifecycle.**

- **3. Describe typical data processing patterns, pipelines and frameworks, e.g. ETL, EtLT, OSEMN.**

- **4. Describe the challenges associated with data processing.**

2. Data collection (L2)

Diversity of Data Sources:

- Data sources vary across many dimensions:
  - **Ownership** — either owned or from third parties; understanding ownership is central, i.e. know what data you have access to and what you can do with it;
  - **Ingestion interface and structure** — how do you get the data and in what form is in;
  - **Volume** —in each step of the pipeline, volume needs to be considered; high- and low- volume are difficult to define and depend on available infrastructures and algorithms;
  - **Cleanliness and validity** —duplicate data, missing or incomplete data, encoding, etc;
  - **Latency and bandwidth of the source** — need to consider internal update requirements and source system limits, speed, timeouts, etc.

Data Selection - Things to Consider:

- Is the author a trustable source that can be contacted?
- Is the data regularly updated?
- Does the data include information about how and when it was acquired?
- Does it seem plausible through observation?

Ingestion Interfaces and Data Structures:

- Examples of ingestion interfaces include:
  - A relational database behind an application, such as PostgreSQL, SQLite or Oracle;
  - A layer of abstraction on top of a system, such as a REST API;
  - An endpoint to a message queue system, such as RabbitMQ;
  - A shared network filesystem, containing logs, CSV files, or other flat files.
- **Examples of data structures include:**
  - JSON from REST API;
  - Well-structured data from a relational database;

    - Semi structured log data;
    - CSV (comma-separated values) datasets;
    - PDF, or other proprietary format, files;
    - HTML, or other semi-structured, files;

Data Encoding:

- Data can be encoded
    - in memory, in specific structures such as objects, lists, arrays;
    - as a self-contained sequence of bytes, for file storage or network transmission, e.g. JSON document.
- 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).
- Most programming languages have built-in support for encoding (and decoding) in-memory data to byte sequences, e.g. Java's java.io.Serializable, Python's pickle, PHPs serialize.
- Useful for transient purposes but in general not adequate in data pipelines — limited to a programming language, reduced interoperability, lower performance, etc.

Data Quality:

- Common problem affecting data quality:
    - Missing data — due to error, specific meaning (e.g. n/a, 0, NULL).
    - Inconsistent values — distinct time zones in time/dates, multiple units (e.g. m, km).
    - Precision problems — rounding decisions may result in fake patterns (e.g. maps).
    - Duplicate values — due to errors, or valid data.
    - Many other: text encoding problems, mislabelled data, incomplete, outdated, etc.
- There is a need to investigating and understand data properties during the data selection phase — often called data investigation or assessment.

Overall View of the Distribution:

- Descriptive statistics are commonly used to begin the investigation.
- Common measures and techniques estimate the:
  - Central tendency, i.e. central, **average value of observations**:
    - Mean (sum divided by the number of items), median (middle value that separates the observations), mode (most frequent value);
  - **Dispersion**, i.e. how much the values vary:
    - Standard deviation, interquartile range (difference between values in upper and lower quartile), difference between the maximum and minimum.
- **Box plots** are a method that graphically depicts most of these descriptive statistics.

Outliers:

- Outliers are items that differ significantly from others.
- It is necessary to understand if these values are exception, but valid cases, or if are errors that need to be removed. Expert domain-knowledge if often necessary.
- Errors resulting in outliers may be the result of:
  - Problems in the data collection procedure;
  - Hardware or software problems in data collection tools;
  - Human mistakes in data recording.
- Outliers may significantly distort descriptive statistics or visualizations.

Missing Data:

- Missing data is an important aspect of data quality that always needs a detailed investigation to determine its origin and impact on following steps.
- Isolated instances of missing data aren't usually a problem, however if the missing data is not randomly distributed or occurs in large numbers globally or in specific variables, the data set will be biased and thus not appropriate for a valid analysis.
- Missing data can also be an indicator of flaws in the data collection process.

Data Quality Summary:

- Investigating the properties of the data at its origin and at different points of the data processing pipeline is a key aspect of a solid data-based project, helping on:
    - Deciding on the data sources to select;
    - Determining possible bias and limitations of the data a priori;
    - Detecting and correcting problems in the pipeline;
    - Framing the conclusions of built products;
- Data quality investigations should rely on multiple methods, from descriptive statistics to exploratory visualization.
- Best practices: clean and validate in the best system to do so; validate often.

Tools for Data Collection:

- SQL for extracting data from databases.
- Custom code for APIs.
- Unix commands to work with external sources, e.g. curl, wget.
- Web crawling platforms:
    - Scrapy (Python)
    - Internet Archive Heritrix (Java)
    - Apache Nutch (Java)

3. Data Preparation (L3)

Data Preparation:

- Real-word data is "messy".
- In practice, 50% to 80% of the time spent in data processing is spent in data preparation tasks.
- Data preparation, often called "data wrangling", captures **activities** like:
    - Understand what data is available;
    - Choose what data to use and at what level of detail;
    - Understand how to combine multiple sources of data;
    - Deciding how to distill the results to a size and shape that enables the follow-up steps.
- After data properties are investigated and understood, a data preparation phase is generally needed to make the data suitable for the follow-up phases.
- Common **data preparation tasks** include:
    - Data **cleaning** — identify and fix data quality issues;
    - Data **transformation** — transform data to improve analysis or manipulation;
    - **Synthesis** of Data — create new attributes derived from existing data;
    - Data **integration** — combine data from different sources;
    - Data **reduction or selection** — eliminate data from the collection.

Data Transformation:

- Data transformation operations can be performed to improve or facilitate data handling in subsequent stages.
- Transformation of data elements include:
    - Normalization of values to a comparable scale;
    - Scaling values to the same range (e.g. 0 to 1 for a follow-up method);
    - Non-linear transformations to deal with skewed distributions;
    - Discretization or binning, which transforms a numeric continuous value into ordered categorical values (e.g. having bins of the same size vs. having bins with the same amount of items)

- Note that all operations over the data introduce layers of distortion and bias that are dependent on assumptions and interpretations. Documentation is key.

Synthesis of Data:

- Combine existing attributes to produce new additional attributes that are more convenient to analyze or use as input in follow-up phases. Examples:
  - a new attribute representing the difference between two timestamps (e.g. duration);
  - the maximum value from a series of numerical attributes;
  - an integrated score that combines several attributes;
  - splitting an existing numerical series in two independent series (e.g. day and night);
  - most important keywords or topics extracted from a textual field;

Data Integration:

- Combine data that originally exists in multiple sources.
- Key task in many projects, e.g. integrate data from multiple databases in an organization; enrich individual records with data from external sources.
- Complexity of data integration tasks differs significantly, from using join operations in a single database management system, to downloading, processing and linking external data sources.
- A central step of many data integration tasks is linking the corresponding records. A task that is easier if unique identifiers are available (rarely!), but of high complexity if based on other information (e.g. names, birth dates, addresses).

Data Reduction or Selection:

- Data reduction or selection may be justified due to several reasons, namely: data is not relevant;
  - data is outdated;
  - data volume exceeds the exiting capacity for processing it;
  - existing precision is excessive, while lower precision is sufficient.
- Techniques for performing data reduction or selection include:

  - data filtering
  - data sampling
  - data aggregation

Data Filtering:

- Data filtering is used to remove data from the dataset, e.g. items with unsuitable values, data that is irrelevant for the scope of the project, outdated data items, data items that must be removed due to legal reasons, etc.
- Data filtering can also be used during development to test the planned approach using a manageable portion of the original collection.
- Data filtering operations are deterministic in nature, e.g. remove data from a given year, remove all references to a specific keyword, remove a specific data attribute, etc.

Data Sampling:

- Data sampling is a non-deterministic process that takes a random subset of the data items of a requested size.
- When designing the sampling method, it is important to ensure that the resulting sample is representative of the complete collection.
- Thus, it is important to analyse the distribution of data attributes before and after the sampling process.

Data Aggregation:

- Data aggregation may be used to reduce excessive detail in data, decisions require:
  - choice of the grouping method, depending of domain-specific criteria;
  - selection of the aggregation operator (e.g. mean, median, min, max, percentile).

Visualization in Data Preparation:

- Data preparation requires a good understanding of the data properties; thus data visualization is usually also used at this stage.

- The role of visualization at this stage is to visually present the result of the execution of different methods, e.g. outlier removal.
- Visualization in data preparation can be applied:
    - Before the application of computational methods to explore the properties of the data, e.g. choose an adequate computational method, detect disparities in data.
    - During the application of computational methods to inspect how data is being processes, e.g. observe intermediate data structures, track the data pipeline execution.
    - After the application of computational methods to evaluate the quality if the results, compare the results between different executions of the pipeline.

Data Pipelines:

- Data pipelines are sets of processes that move and transform data from various sources to various destinations where new value can be derived.
- Complexity of data pipelines varies widely, depending on size, state, structure of the data sources as well as the needs of the project.
    - Simple example: obtain data from a REST API and load it into a relational database;
    - Complex example:
        - obtain data from multiple web pages at regular intervals;
        - parse HTML and extract main keyword from each page;
        - automatically identity main trending topics using a previously trained machine learning model;
        - update model with new data (keep an archive of previous models);
        - integrate topics and web pages in a cloud-based storage linked to a live dashboard.

Characteristics of a Data Pipeline:

- A data pipeline is a software system; thus, general software best practices apply. Those highlighted here are particularly relevant in the context of data pipelines.

    - **Reliable** — work as expected even in face of adversity (software, hardware, or human faults).
    - **Scalable** — cope with increased load (in volume, traffic, complexity) in a manageable way (e.g. adding resources, distributing load).
    - **Maintainable** — evolve through time and teams (reduce complexity, make changes easier).
- Data collection and preparation is usually an ad-hoc and exploratory process, easily leading to a dispersion in threads and activities. Adoption of pipeline management systems (e.g. Makefiles) and a complete and detailed documentation is key.
- Treating data processing pipelines as software makes them more maintainable, versionable, testable and collaborative.

Makefiles:

- Makefiles are used to automate software build processes, by defining targets and rules to execute. The underlying abstraction is of a dependency graph, where tasks depend on the execution of other tasks.
- Make language agnostic, and implementations exist for most operating systems.
- Can be used to document and setup data pipelines.

4. Data Processing (L4)

Data Models:

- Data models have a role part in software development, not only how software is implemented but also on how we think about the problem.
- Defining abstraction layers is a common pattern in software architectures, where each layer hides the complexity of the layers below.
- When thinking in data models, each layer hides the data complexity of previous layers, and provides a cleaner data model.
- Different data models make different operations faster/slower, natural/awkward, possible/impossible.
- Models discussed: relational model, document model, and graph-based models.

Relational Model:

- The relational model is the best-known and most dominant data model.
- In the relational model data is organized in relations (called tables in SQL), where each
- relation is an unordered collection of tuples (rows in SQL).
- The NoSQL (Not Only SQL) movement emerged in the 2010s as a need to provide greater scalability, specialized query operations, and a more flexible and expressive data model.
- The "object-relational mismatch" (or impedance mismatch) is an expression used to represent a common criticism to the SQL data model — the need of a translation layer between "tables, rows, and columns" and the "objects" used in object-oriented programming.

Document Model:

- In the document models, data is organized as single instances (documents).
- In document databases, nested records are stored together with their parent records.
- Also known as document stores, document-oriented databases.
- Advantages of document data models are:
- schema flexibility;
- better performance due to locality (i.e. related data is stored together);

- for some applications, the data model is closer to the application's data structures.
- The relational model is better suited to support joins, N-1, and N-N relationships.

Schema Flexibility:

- Most document databases do not enforce any scheme on the data in documents.
- No schema (schemaless) means that arbitrary keys and values van 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!
- This solution is advantageous 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.
- Where the schema is enforced is the key issue — schema-on-read (the structure is enforced when data is read, at code level); schema-on-write (the structure is enforced by the database, the traditional relational approach).

Property Graph Model:

- When the application has mostly one-to-many relationships (i.e. tree-like structures) or no relationships between records, the document model is appropriate.
- If many-to-many relationships are very common, and the connections within the data are complex, it is more natural to model data as a graph.
- A graph has to kinds of objects: vertices (also known as nodes or entities) and edges (also known as relationships or arcs).
- Graphs are not limited to homogeneous data, an advantage of the graph model is the fact that it provides a consistent way of storing completely different types of data in a single datastore (e.g. people, places, events).
- Typical examples, include social graphs, web graph, transportation networks.

Triple-Store Model:

- Similar to the property graph model.
- Important because there are various relevant tools and languages for triple-stores.

- In a triple-store, all information is stored in the form of three-part statements: (subject, predicate, object). For example: (Maria, enrolled, PRI); (PRI, course, M.EIC).
- The subject of a triple is equivalent to a vertex in a graph.
- The object is either, a value (primitive datatype) or another vertex in the graph.
- RDF is a data model for specifying data subject-predicate-object (triples) statements. SPARQL is the querying language for triple-stores using the RDF data model.

Data Storage Summary:

- Data models are central in data processing.
- The relational data model is the most widely used approach.
- Non-relational solutions have emerged in recent years under the "NoSQL" umbrella.
- Two main alternative directions — document databases and graph databases.
    - With document databases, data is mapped to self-contained documents and relationships between documents are rare.
    - With graph databases, data is separated in independent units and, potentially, everything is related to everything.
- These three models are widely used, and each is good in its respective domain.
- There is no one-size-fits-all solution, that's why different systems exist. Also, database systems support features from "competing" models, e.g. JSON support in relational database systems.

Managing Data Pipelines:

- A data processing project is typically a fragmented process, involving multiple operations, data storages, files, folders, formats, etc.
- Managing these processes and workflows is a requirement. The goal is to produce "software artifacts" that are **maintainable**, **versionable**, **testable**, and **collaborative**.

Make:

- Make is a 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).
- Make was originally built, and is still a central piece, in software building processes.

- It can generally be used in scenarios where the execution workflow can be encapsulated as a set of dependencies and execution rules.
- Make can automatically decide what to execute based on direct or indirect dependencies to files that have changed.
- Make is language agnostic and follows a declarative programming paradigm.
- Use cases: software building, scientific publishing (figures, tables, LaTeX files, etc), data pipelines.

Makefiles:

- Makefile is a text file where Make execution rules and dependencies are defined.
- The default filename is "Makefile", with no extension and starting with an uppercase.
- A makefile consists of a series of rules.
    - Each rule defines a target name (or list) and an optional list of prerequisites.
    - Additionally, a rule can have a list of commands to execute (recipe).
    - Recipe commands must be indented with TABs.
- To build a specific target, "make <target>" is used.

Data Presentation:

- Data presentation is an important task in data projects.
- Depending on the audience, presentation can differ greatly.
- Presentation can simply use tables to effectively communicate the key aspects.
- Or use complex visualizations, static or interactive.
- We saw that data visualization can be an important tool in data preparation, e.g. to investigate data or evaluate data quality issues. In this case, the audience is mostly specialized.

Data Visualization:

- Data visualization (data viz) refers to algorithmically drawn, easy to regenerate, and aesthetically simples visualizations. As opposed to infographics, which are manually drawn and aesthetically more complex visualizations.

- Data visualizations can be classified according to their complexity considering the number of dimensions they represent, i.e. the number of discrete data types visually encoded in the diagram.
- Visualization can also be divided in two categories, each with a different purpose: exploration and explanation.
    - 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.
- In the context of PRI,
    - we are interest in simple (but insightful), static visualizations;
    - that help us understand and highlight relevant characteristics of the data;
    - and are programmatically obtained, i.e. use scripts to quickly generate the visualizations.
- For other end-users, presentation solutions will depend, and can range from static images (e.g. in internal documentation), to publicly available websites (e.g. intranet), to interactive applications (e.g. data journalism).

Elements of Visualization:

- Common data visualization solutions include:
- **charts**, suitable for numeric data, especially when comparing data sources or groupings;
- **time-series**, or timelines, help represent observations over time;
- **maps**, focus on geographic properties of data;
- **interactive**, where control is given to the user to explore different paths or options;
- **words**, representing highlights (e.g. most frequent) of textual descriptions;

Questions:

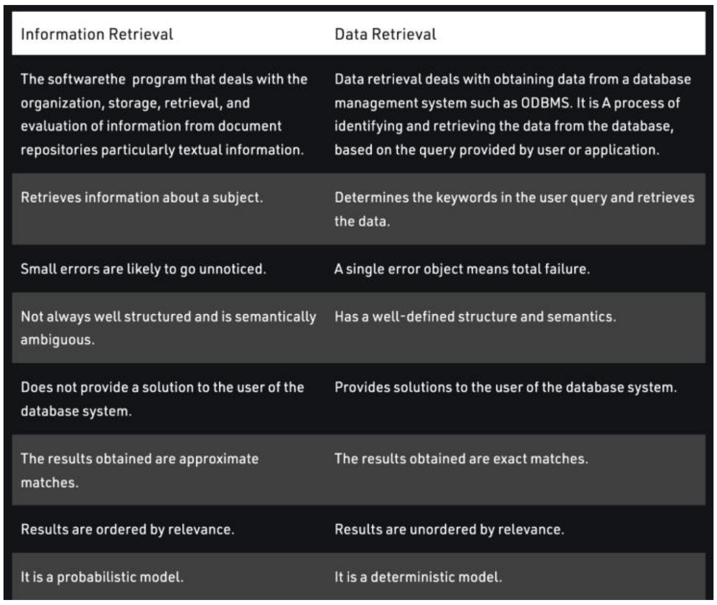
- **5. Identify and describe challenges and techniques related to data cleaning, data preparation, and data presentation.**

- **6. Describe the importance of data pipelines and how Makefiles can be used to implement them.**

Data collection and preparation is usually an ad-hoc and exploratory process, easily leading to a dispersion in threads and activities. Adoption of pipeline management systems (e.g. Makefiles) and a complete and detailed documentation is key. Treating data processing pipelines as software makes them more maintainable, versionable, testable and collaborative.

5. IR tasks and systems (L5)

Information Retrieval vs. Data Retrieval:

- Information Retrieval distinguishes itself from Data Retrieval in two central aspects:
    - Not limited to exact matching (e.g. multiple words, synonyms)
    - Is ordered by relevance, i.e. importance to the user's query (a central concept in IR).
- 

Information Retrieval	Data Retrieval
The softwarethe program that deals with the organization, storage, retrieval, and evaluation of information from document repositories particularly textual information.	Data retrieval deals with obtaining data from a database management system such as ODBMS. It is A process of identifying and retrieving the data from the database, based on the query provided by user or application.
Retrieves information about a subject.	Determines the keywords in the user query and retrieves the data.
Small errors are likely to go unnoticed.	A single error object means total failure.
Not always well structured and is semantically ambiguous.	Has a well-defined structure and semantics.
Does not provide a solution to the user of the database system.	Provides solutions to the user of the database system.
The results obtained are approximate matches.	The results obtained are exact matches.
Results are ordered by relevance.	Results are unordered by relevance.
It is a probabilistic model.	It is a deterministic model.

Information Retrieval:

- In computer science, the field that is most involved in search problems and developments is [Information Systems > Information Retrieval], and includes sub-areas such as:
- Document representation
- Information retrieval query processing Users and interactive retrieval
- Retrieval models and ranking
- Retrieval tasks and goals
- Evaluation of retrieval results
- Search engines architectures and scalability

Defining Information Retrieval:

- "**Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information**", Salton (1968)

    - General definition that can be applied to many types of search applications.
    - Primary focus since the 50s has been on text and documents.
- "Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)", Manning et al. (2008)

Different from Databases:

- "A database management system is a software system that enables the creation, maintenance, and use of large amounts of data" (Abiteboul et al., 1995)
- Information retrieval and database systems have a lot in common.
- But emphasize different aspects of information management.
    - Databases contains highly structured data, queried using formal query languages, and the results are set-based — results are true or false for a given condition.
    - Information retrieval systems support interactive processes, support natural language queries, and results are rank-based — results are relevant for a given information need. Not just text matching!

Information Retrieval Tasks:

- Information retrieval applications and research are typically structured around tasks.
- Tasks address specific contexts, and specific information access problems.
- Examples of classical **information retrieval tasks**:
    - **Ad-hoc search**, find relevant documents for an arbitrary text query.
    - **Filtering**, identify relevant user profiles for a new document.
    - **Classification**, identify relevant labels for documents.
    - **Question answering**, give specific answer to a question.
- Yearly evaluation initiatives with classical and novel tasks at TREC.

The Text Retrieval Conference (TREC) is an annual event that focuses on information retrieval research and experimentation. TREC provides a framework for evaluating different retrieval tasks using benchmark datasets.

Central issues in information retrieval:

Relevance:

- The concept of relevance is a key issue in information retrieval.
- Many factors impact a person's decision of what is relevant, e.g. task, context, previous knowledge, etc.
- Simple definition of relevance,
    - a relevant document contains information that satisfies the information need of the user that submitted a query to the search engine

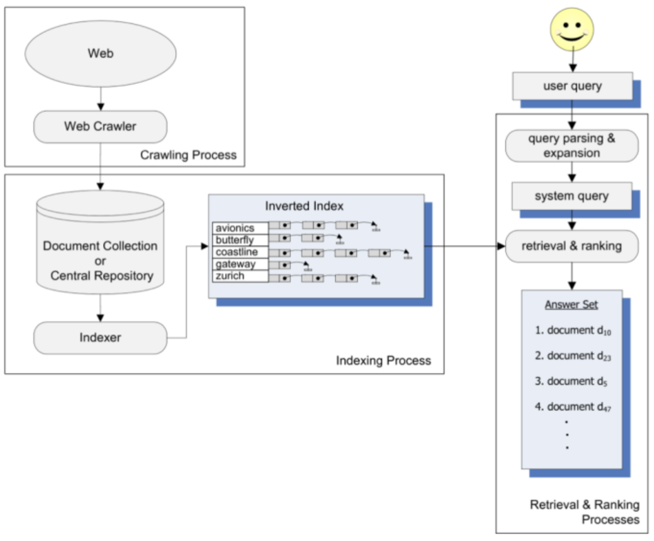
Evaluation:

- Evaluation setups are based on the use of annotated test collections of documents, queries, and relevance judgements.
- Evaluation in information retrieval dates to the 1960s (Cranfield paradigm).
- IR evaluation methodologies have been adopted by many fields.
    - Recall and precision are two examples of widely used measures.
- Information needs drive the search process and result from the underlying task users' have.

Architecture of information retrieval systems:

Information Retrieval System Modules:

1. **Crawling Process**: Web --> Web Crawler
2. **Indexing Process**: Document Collection --> Indexer --> Inverted Index
3. **Retrieval and Ranking Process**



Questions:

- **1. What is the difference between information retrieval and data retrieval?**

- **2. Give examples of IR and data retrieval systems.**

Information Retrieval systems (search systems):

- Google
- Bing
- Elasticsearch
- Apache Solr

Data Retrieval systems (data bases):

- SQL Databases (e.g., MySQL, PostgreSQL)
- NoSQL Databases (e.g., MongoDB, Cassandra)

- **3. Give some examples of retrieval tasks evaluated in TREC.**

- **4. What are the modules of an IR system?**

Completed by Sofia Merino Costa
up202300565

6. IR concepts (L8)

[Concepts: document, information need, relevance, bag of words, inverted index, postings list, term pre-processing.]

Documents:

- Documents can be web pages, email messages, books, news articles, research papers, computer files (PDF, Word, text), text messages, etc.
- Common properties:
  - Significant text content.
  - Some structure, e.g. title, body, date, tags, author.

Boolean Model:

- In the Boolean Retrieval Model queries are represented in the form of a Boolean expression of terms.
- E.g: [Brutus AND Caesar AND NOT Calpurnia]
  - => 110100 AND 110111 AND 101111 = 100100
- The model views each document as a set of words.
- **Bag of Words (BoW) model**, where the exact ordering of terms in a document is ignored and only their presence is considered.
- Documents are represented in a term-document matrix.
- Queries are specified as Boolean expressions.
- The Boolean model predicts if documents are either relevant or non-relevant.
- No ranking is provided.

Index Construction: Choosing the document unit for indexing and the index granularity are important first steps. There is a precision / recall trade-off in this decision.

Tokenization: A token is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing.". "A type is the class of all tokens containing the same character sequence.".

Stop Words:

- Stop words are words considered of little value in helping select documents in the retrieval process, e.g. a word that exists in all documents.

- The strategy to determine stop words is by looking the collection frequency of a term (cf), i.e. the total number of times a term appears in the document collection.
- A **stop list** is a (commonly) hand-picked list of terms to be discarded during the indexing process.

Token Normalization:

- "**Token normalization** is the process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences of the tokens."
- Some forms of normalization commonly employed are: accents, capitalization, and stemming and lemmatization for dealing with different forma of a word (e.g. watch, watches, watching).
- **Stemming** refers to a heuristic process that chops off the ends of words in the hope of reducing inflectional forms.
- **Lemmatization** refers to the process of reducing inflectional forms by using vocabularies and the morphological analysis of words to find its lemma.

Collection: A collection can be defined as a bank in which all the documents are stored. It is the equivalent of a database, in IR systems. It consists of a group of documents, over which retrieval is performed.

Term: A term is the indexed unit of an incidence matrix. A term is a (perhaps normalized) type that is included in the IR system's dictionary.

Information Need:

- An information need refers to the user's requirement or desire for specific information to satisfy a particular purpose or goal.
- Key concept in information retrieval.
  - The goal of an effective information retrieval system is to retrieve all relevant documents, without retrieving non-relevant ones.
  - Intuitive concept for humans that needs to be formalized for implementing measures.

Index: Data structure that facilitates efficient and fast retrieval of documents based on the terms they contain.

Query: Formal request or expression of information needs made by a user to a search engine or information retrieval system.

Relevance: A document is relevant if it addresses the stated information need, not because it just happens to contain all the words in the query.

Ranked Retrieval:

- With the Boolean model, a document either matches or does not match a query.
- In large document collections this is not feasible, thus it is essential for a search system to rank-order the documents.
- Note that there are scenarios where recall is **determinant** — i.e., all documents need to be analysed (e.g., patent search), and thus Boolean search is used.

Relevant result in a results list: A results list is the list returned by an information retrieval tool, for a given query. Unless the tool is perfect, only some of these are perceived by the user as containing information of value concerning their personal information need, which originated the query given to the tool.

Inverted Index: Indexes were created to avoid having to scan through the entirety of a collection to verify whether a term was present or not. Using an incidence matrix, or term-document matrix, we can create a matrix with documents and terms in its rows and columns, which takes the value of 1 if the term exists in said document, and 0 if it doesn't. This naive approach does not scale well. It was discovered that most of the matrix would consist of 0's. Thus, the inverted index idea originated. An inverted index is one in which the index points to the locations in a file where the term occurs. Meaning, if we have an inverted index for the word 'Cesar', this index will point to a list of locations, which represent the locations in which this term occurs.

Vocabulary: A dictionary of terms, or lexicon, is usually called a vocabulary. This is a dictionary containing all the terms in each collection (a bank of records, set of documents). This contains all the terms that can be retrieved.

Postings List: Complementary to the vocabulary, a postings list is a list in which it is stored, for each term in the vocabulary, which documents the term appears in, and later, often, the positions in which it occurs.

Term Pre-Processing: process of preparing and cleaning textual data to enhance its suitability for natural language processing (NLP) and information retrieval (IR) tasks.

Bag of Words: In this view of a document, known in the literature as the bag of words model, the exact ordering of the terms in a document is ignored but the number of occurrences of each term is material (in contrast to Boolean retrieval). We only retain information on the number of occurrences of each term. Thus, the document "Mary is quicker than John" is, in this view, identical to the document "John is quicker than Mary". Nevertheless, it seems intuitive that two documents with similar bag of words representations are similar in content.

Parametric and Zone Indexes:

- Although we have considered documents to be a simple sequence of terms, most documents have additional structure (e.g. email message). Additionally, metadata is often associated with a document (e.g. date, authors, title).
- **Parametric indexe**s are inverted indexes built for specific parameters, or fields, that support parametric search (e.g. "all documents from author Z containing word Y").
- **Zones** are a similar concept applied to arbitrary free text (e.g. portion of a document). For example, a document's abstract can be associated to a specific zone index.

Ranked Boolean Retrieval:

- Zones (or 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.
- Zone weights can be specified by an expert (commonly the end user) but are usually learned by the system based on training examples. This method is known as "machine-learned relevance" or "learning to rank".

Questions:

- **1. What is… a document, a collection, a term, a bag of words?**
- **2. Define stemming.**
- **3. What is… an inverted index, a vocabulary, a postings list?**
- **4. What is… an information need, a query, a results list?**
- **5. What is a relevant result in a results list?**

7. NLP in Information Retrieval (L4)

Natural Language Processing (NLP): is a field of artificial intelligence that focuses on the interaction between humans and computers through natural language.

In **Information Retrieval**, NLP plays a **key role** in improving the effectiveness of search and retrieval systems.

- Extraction of structured information from unstructured text, making it easier to index and search.
- Help in understanding user queries and documents, improving relevance ranking.
- Extraction of entities, relations, and sentiment from text, enabling richer retrieval experiences.

Key NLP Concepts in IR:

- **Tokenization:**
  - Tokenization is the process of breaking text into individual words or tokens.
- **Part-of-Speech (POS) Tagging:**
  - POS tagging assigns grammatical categories (e.g., noun, verb, adjective) to each word in a sentence.
- **Named Entity Recognition (NER):**
  - NER identifies and classifies entities such as names of people, places, organizations, and dates in text.
- **Syntactic Analysis:**
  - Syntactic analysis involves parsing sentences to understand their grammatical structure.

NLP Terminology:

- **Stemming and Lemmatization**
  - Stemming reduces words to their root form (e.g., "jumping" to "jump"), aiding in query expansion and document matching.
  - Lemmatization is a similar process that reduces words to their base or dictionary form (e.g., "better" to "good").
- **Semantic Analysis**

    - Semantic analysis explores the meaning of words and phrases, enabling the understanding of context.
    - It's essential for query understanding, disambiguation, and identifying synonyms.
- **Sentiment Analysis**
    - Sentiment analysis determines the sentiment (positive, negative, neutral) expressed in text.
    - In IR, sentiment analysis can be used for personalized content recommendation and sentiment-based filtering.
- **Information Extraction**
    - Information extraction involves identifying structured information (e.g., dates, events, relationships) within unstructured text.
    - It's valuable for creating structured databases from textual sources.

NLP Approaches:

- **Rule-Based Approaches:**
    - Rule-based NLP relies on predefined rules and patterns to analyze and manipulate text.
    - These rules are designed by linguists or experts.
    - Example: Email Address Extraction
        - Rule: Identify sequences of characters with "@" and "." to capture email addresses.
    - This approach is precise but may struggle with complex or evolving language patterns.
    - Is also costly and hard to scale.
- **Statistical Approaches:**
    - Statistical models use probabilistic techniques to analyse language based on the statistical properties of large corpora of text.
    - Example: Language Model for Text Prediction
        - Model: Predict the next word in a sentence based on the probability distribution of word sequences.

    - Statistical approaches excel in tasks like speech recognition and machine translation.

Text Representation:

- Text representation is a necessary first- step in an NLP process.

Frequency-Based Representations:

- Bag of Words (BoW)
- Term Frequency-Inverse Document Frequency (TF-IDF)
- N-grams

Semantic Representations:

- Word Embeddings
- Document Embeddings
- Topic Models

Structure-Based Representations:

- Syntax-Based Representations
- Graph-Based Representations

8. Vector (space) model (L8)

Ranked Retrieval:

- With the Boolean model, a document either matches or does not match a query.
- In large document collections this is not feasible, thus it is essential for a search system to rank-order the documents.
- Note that there are scenarios where recall is determinant — i.e. all documents need to be analysed (e.g., patent search), and thus Boolean search is used.

Term weighting: Quantified importance of a term in a document relative to its rarity across the entire collection.

Bag of Words (BoW):

- BoW represents text as an unordered collection of words, disregarding grammar and word order. BoW simplifies text into term frequency counts for each document.
- Example and representation for the document:
  - "'Good dog, good dog!', said the quick brown fox."
  - {"good": 2, "dog": 2, "said": 1, "the": 1, "quick": 1, "brown": 1, "fox": 1}
- **Advantages:** simplicity, efficiency, and word independence.
- **Disadvantages:** loss of context, semantic gaps, and sparsity.

Term Frequency-Inverse Document Frequency (TF-IDF):

- Assigns a weight to each term in a document based on its frequency in the document and rarity across the entire corpus.
- Example and representation for the document:
  - "Good dog, good dog!', said the quick brown fox."
  - {"Good": 0.2, "dog": 0.3, "said": 0.15, "quick": 0.25, "brown": 0.3, "fox": 0.3}
- **Advantages**: capture term importance, discriminates between term, and flexibility.
- **Disadvantages**: loss of context, semantic gaps, no word order, and sparsity.

N-Grams:

- N-grams represent text by considering sequences of n contiguous words.
- A "bag of n-grams".
- Example and representation for the document (with bigrams, n=2):
  - "'Good dog, good dog!', said the quick brown fox."

    - {"'Good dog,", "dog, good", "good dog!’,", "dog!’, said", "said the", "the quick", "quick brown", "brown fox."}
- **Advantages**: contextual information, phrase detection, and improved semantics.
- **Disadvantages**: increased dimensionality, and size sensitivity.

Vector Based:

- **Word Representation:**
    - Map each word into a vector with the size of the vocabulary (i.e. one-hot encoding).
    - "'Good dog, good dog!’, said the quick brown fox.”
    - Vocabulary: [brown, dog, fox, good, quick, said, the]
        - good: [0, 0, 0, 1, 0, 0, 0]; dog [0, 1, 0, 0, 0, 0, 0]
- **Document representation:**
    - Using word frequencies: [1, 2, 1, 2, 1, 1, 1]
    - Using normalized unitary vectors (later topic).

Word Embeddings:

- **Word Embeddings** represent words as dense vectors in a continuous space, capturing semantic relationships.
- **Document Embeddings** represent entire documents as a single dense vector.
- **Advantages:**
    - Semantic Similarity: capture semantic relationships, enabling measurement of word similarity and analogy tasks (e.g., "king" - "man" + "woman" = "queen").
    - Reduced Dimensionality: typically have lower dimensionality compared to one-hot encodings, making them computationally more efficient.
- **Challenges:**
    - Data Intensity: training high-quality word embeddings requires large text corpora and computational resources.
    - Interpretable Features: interpreting the components of word embeddings can be challenging, as they are learned automatically.

Vector Space Model:

- The representation of a set of documents as vectors in a common vector space is known as the vector space model and is fundamental to number Information Retrieval operations.
- In a nutshell, each document is represented as a vector, with a component vector for each dictionary term. TF-IDF weights are used as component weights.
- 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:**
    - To quantify the similarity between two documents in this vector space, the cosine similarity of the vector representations of the two documents.
    - In other words, the similarity between two documents is given by the cosine of the angle between the two vector representations of the documents.
    - This approach compensates the effect of document length.
    - $$\mathrm{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|},$$
- **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.

Questions:

- **What is the bag of words model for a document?**

[described above]

- **What is… term frequency, collection frequency, document frequency, inverse document frequency?**

Term frequency (TF):

- The next step is not to consider only the presence (or not) of a query term in a zone, but the number of mentions of the term (i.e., term frequency).
- The simplest form of weighting terms differently is simply to assign the weight of a term to the term's frequency.
- The term frequency of a term in a document is **denoted** $tf_{t,d}$
- In the bag of words model, the ordering is ignored but the number of occurrences of each term is key (in contrast with Boolean retrieval).

Collection Frequency (CF):

- It refers to the number of documents or texts in a collection (corpus) that contain a specific term.
- Collection frequency is a crucial metric in the calculation of the inverse document frequency (IDF) component of the TF-IDF (Term Frequency-Inverse Document Frequency) weighting scheme.

Document Frequency (DF):

- Raw term frequency suffers from a problem: all terms are considered equally important when assessing a query, when in fact some terms are of little use in determining relevance.
- For example, in a collection of thesis dissertations, the term "dissertation" is less likely to be of value since this term probably exists in every document.
- An important measure to incorporate the discriminative power of a term in a collection is the document frequency of a term (df), i.e. the number of documents that contain a term.

Inverse Document Frequency (IDF):

- The document frequency of a term is incorporated in the weight of a term by using the concept of inverse document frequency (idf).
- IDF = log(N/df t), where N is the total number of documents in the collection.
- $$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}.$$
- The **rarer** the term in a collection, the **higher its IDF**.

- <u>**How do you calculate tf-idf weights?**</u>

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t.$$

→ **tf-idf t,d** 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.

- **<u>How do you rank documents in the vector model?</u>**

The ranking of documents is based on the (cosine) similarity between the query vector and the document vectors.
The fundamental idea is that documents that are more like the query, are ranked higher.
The higher the similarity score, the higher the document is ranked.

...Exercises: look at Exercises 6.8, 6.9, 6.10, 6.11, 6.15, 6.16, 6.17 and Examples 6.2, 6.3, 6.4.

9. Evaluation (L7)

[Precision, recall, P-R curves, MAP, reference collections, relevance judgements.]

Evaluation of Information Retrieval Systems:

- Evaluation depends on the task, collection, information need type.
- Collections: news articles, web pages, scientific articles, etc.
- Different information needs:
  - What is the homepage of the Dept. of Informatics Engineering at FEUP?
  - Who are the teachers of information systems courses at FEUP?
  - Open-source information retrieval frameworks?
- Informational vs. navigational; Number of relevant documents; Answers needed.

Evaluation of Information Retrieval Systems

- Evaluation is at the heart of Information Retrieval.
- Evaluation is important to:
  - Understand the use of a system by its users.
  - Make decisions on new designs and features to implement.
- A primary distinction must be made between **effectiveness** and **efficiency**.
  - **Effectiveness** measures the ability of a search system to find the right information.
  - **Efficiency** measures how quickly a search system provides an answer.
- User satisfaction encapsulates these and other aspects (ux, coverage, effort, etc).

Information Retrieval System Evaluation:

- To measure the effectiveness of a search system in the standard way, we need three things:
  - A document collection;
  - A test suite of information needs, expressible as queries;
  - A set of relevance judgements, typically a binary assessment of either relevant or non- relevant for each query-document pair.
- The standard approach to IR system evaluation revolves around the notion of relevant and non-relevant documents.

- With respect to a user information need, a document in the test collection is given a binary classification as either relevant or non-relevant (gold standard or ground truth).

Test Collection Components:

- A document collection
- A test suite of information needs, expressible as queries
- A set of relevance judgments, standardly a binary assessment of either relevant or nonrelevant for each query-document pair

Information Need:

- Relevance is assessed relative to an information need, not a query.
- An information need might be:
    - Information on whether drinking red wine is more effective at reducing your risk of heart attacks than drinking white wine.
- This might be translated into a query such as:
    - [wine red white heart attack effective]
- A document is relevant if it addresses the stated information need, not because it just happens to contain all the words in the query. This distinction is often misunderstood in practice, because the information need is not clear.

Relevance:

- Key concept in information retrieval.
- The goal of an effective information retrieval system is to retrieve all relevant documents, without retrieving non-relevant ones.
- Intuitive concept for humans that needs to be formalized for implementing measures.

Manifestations of Relevance:

- **System relevance**: relation between a query and information objects (texts) in the file of a system as retrieved.
- **Topical relevance (aboutness)**: relation between the subject or topic expressed in a query, and topic or subject covered by retrieved texts
- **Cognitive relevance (pertinence)**: relation between the state of knowledge and cognitive information need of a user, and texts retrieved (e.g., cognitive correspondence, informativeness, novelty, information quality).

- **Situational relevance (utility)**: relation between the situation, task, or problem at hand, and texts retrieved (e.g., usefulness in decision making, appropriateness of information in resolution of a problem, reduction of uncertainty).
- **Affective relevance (satisfaction)**: relation between the intents, goals, and motivations of a user, and texts retrieved (e.g., satisfaction, success, accomplishment)

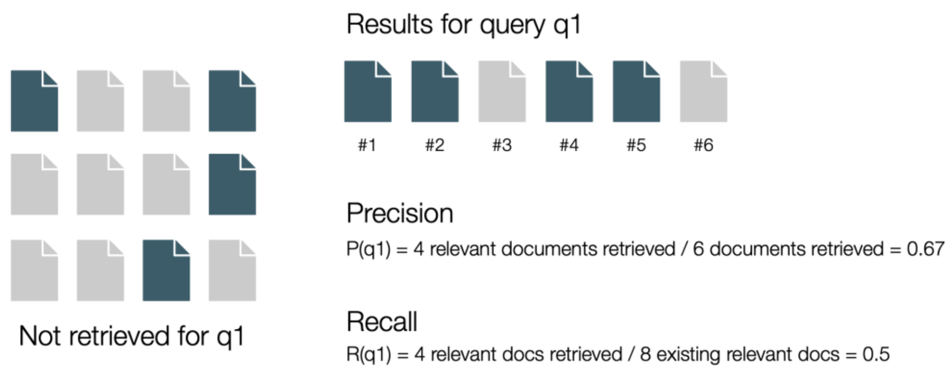
The Cranfield Paradigm:

- Evaluation of Information Retrieval systems is the result of early experimentation initiated in the 50's by Cyril Cleverdon.
- The insights derived from these experiments provide a foundation for the evaluation of IR systems.
- These experiments culminated in the metrics of Precision and Recall.
- 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.

Evaluation of Unranked Retrieval:

- The two most frequent and basic measures for information retrieval effectiveness are Precision and Recall.
- Precision is the fraction of retrieved documents that are relevant.
    - Precision (P) = #(relevant items retrieved) / #(retrieved items)
- Recall is the fraction of relevant documents that are retrieved.
    - Recall (R) = #(relevant items retrieved) / #(relevant items)
- Precision and Recall are set-based measures.

Precision and Recall (Example):



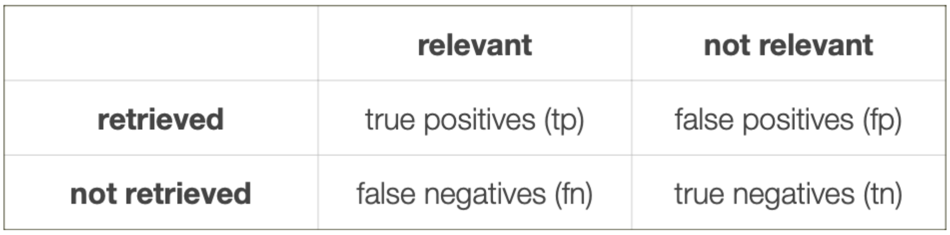
Precision

P(q1) = 4 relevant documents retrieved / 6 documents retrieved = 0.67

Recall

R(q1) = 4 relevant docs retrieved / 8 existing relevant docs = 0.5

Contingency Table:



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

- Precision = true positives / (true positives + false positives)
- Recall = true positives / (true positives + false negatives)
- Accuracy, the fraction of classifications that are correct (not useful for IR).
  - Accuracy = #(true positives + true negatives) / #(tp + fp + fn + tn)

F measure:

- A measure that trades-off Precision versus Recall is the F measure (or F score), which is the weighted harmonic mean of precision and recall.
- Often criticized because it is opaque to interpretation.

Evaluation of Ranked Retrieval:

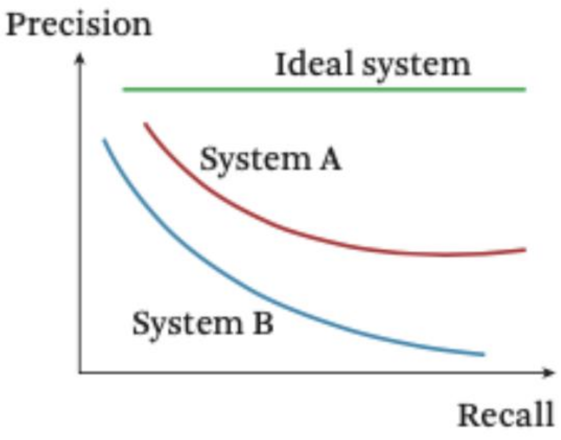
- Precision and Recall are computed over unordered sets of documents.
- These measures need to be extended to evaluate the ranked lists of results common in search engines.
- In ranked retrieval contexts, appropriate sets of retrieved documents are naturally given by the top k retrieved documents. For each set, Precision and Recall values can be computed.
- These values can be plotted to obtain a precision-recall curve.

Interpolated precision is a metric used to evaluate the precision of a search or retrieval system at different recall levels. Precision measures the relevance of retrieved documents among all the documents retrieved. Interpolated precision involves interpolating precision values at various recall levels.

Precision-Recall Curves:

- Consider the ordered set of relevant (R) and non-relevant (N) results from a search system A:
  - Sa = R R N R N N R N R N

- In this ranking, the first result is relevant and corresponds to 20% of all (available) relevant documents.
    - We say that we have 100% precision at 20% recall.
- At position 4, three documents out of four are relevant, and three documents of a total of five relevant document have been retrieved.
    - We say that we have 75% precision at 60% recall.
- Using this data, we can plot a precision-recall curve.



Precision at k (P@k):

- In the case of web search, most users do not require high recall.
- What matters are high quality results on the first page. This leads to measuring precision at fixed low levels of retrieved results.
- For example, "precision at 5" (P@5) or "precision at 10" (P@10).
- Considering the following ranking for a given query:
    - R R N NR NR R R R
    - P@5 = 0.6;
    - P@10 = 0.7

Mean Average Precision:

- Average Precision (AvP) provides a single-figure measure of quality across recall levels for a single query.
- For a single information need, average precision is the average of the precision value obtained for the set of top k documents existing after each relevant document is retrieved.

- Given a set of queries, the Mean Average Precision (MAP) is the mean over the AvP values. This is one of the most used measures in IR.

User-Centered and Online Evaluation:

- Instead of using standard collection, systems may be evaluated using real user interaction data. This requires a working system or prototype with a user base.
- In user-centered evaluation, different techniques may be used
    - Observational studies, observe the user in controlled sessions;
    - Observe server-side query logs, to measure performance;
    - Use client-side logging tools, to study interaction patterns;
- Online, large-scale evaluation experiments can be made with A/B tests. Common practice in industry - run multiple experiments with different versions of the system.

Questions:

- **1. What is… precision, recall, interpolated precision?**

Precision and Recall are set-based measures.

Precision is the fraction of retrieved documents that are relevant.

Precision (P) = #(relevant items retrieved) / #(retrieved items)

Recall is the fraction of relevant documents that are retrieved.

Recall (R) = #(relevant items retrieved) / #(relevant items)

Interpolated precision is a metric used to evaluate the precision of a search or retrieval system at different recall levels. Precision measures the relevance of retrieved documents among all the documents retrieved. Interpolated precision involves interpolating precision values at various recall levels.

- **2. What is… precision at k, R-precision?**

[described above]

- **3. Name the components of a test collection.**

[described above]

- **4. Why is a set of relevance judgements considered a "ground truth" for IR?**

With respect to a user's information need, a document in the test collection is given a binary classification as either relevant or non-relevant (gold standard or ground truth).

It represents a definitive evaluation of the relevance of documents to a user's information need. These judgments are typically created by human assessors who manually assess and label documents as relevant or non-relevant based on their relevance to a specific query or information need. **Relevance judgements represent the best approximation of relevance/ground truth based on human judgment**.

- **5. Draw a precision-recall curve for capturing the evolution of precision in the ranked list of results for a query.**

- **6. What is an average 11-point precision-recall graph for a set of queries?**

For each information need, the interpolated precision is measured at the 11 recall levels of 0.0, 0.1, 0.2, . . . , 1.0.
For the precision-recall curve in Figure 8.2, these 11 values are shown in Table 8.1. For each recall level, we then calculate the arithmetic mean of the interpolated precision at that recall level for each information need in the test collection. A composite precision-recall curve showing 11 points can then be graphed.

- **7. What is MAP, and do you calculate it for a set of queries in a test collection?**

[described above]

Exercises: look at Exercises 8.1, 8.4, 8.8, 8.9.

Completed by Sofia Merino Costa
up202300565

Completed by Sofia Merino Costa
up202300565

10.Web search (Manning PDF – Chapter 19)

[Web information needs, the bow-tie model, web search vs enterprise search, multimedia content, ranking functions and ranking signals.]

Questions:

- **1. What are informational, transactional and navigational information needs?**

Informational needs: Informational queries seek general information on a broad topic, such as leukaemia or Provence. There is typically not a single web page that contains all the information sought; indeed, users with informational queries typically try to assimilate information from multiple web pages.

Transactional needs: A transactional query is one that is a prelude to the user performing a transaction on the Web – such as purchasing a product, downloading a file or making a reservation. In such cases, the search engine should return results listing services that provide form interfaces for such transactions.

Navigational needs: Navigational queries seek the website or home page of a single entity that the user has in mind, say Lufthansa airlines. In such cases, the user's expectation is that the very first search result should be the home page of Lufthansa. The user is not interested in a plethora of documents containing the term Lufthansa; for such a user, the best measure of user satisfaction is precision at 1.

- **2. Name some differences between web search and enterprise search.**

Web searches are designed for end users.

Enterprise searches often need to support both end users and client systems.

End users will require UI development. Other internal applications may want to query and surface results within their own UI by using search APIs. Web search usually only deals with websites accessible over http, but many enterprises want to be able to search various file shares or databases. That's why most enterprise solutions will include connectors for things like SharePoint, Domino, or connectivity to systems over NFS or SMB.

- **3. How do you index images?**

To index images in IR, extract features like LBP or HOG, preprocess images (normalize, resize), represent features as vectors, use inverted indexing, employ hashing (LSH, binary), incorporate metadata, evaluate with metrics like precision and recall, and integrate with text-based indexing for multimodal retrieval.

- **4. Give examples of ranking signals used by search engines.**

Ranking Signals:

Signals can also be distinguished according to other dimensions.

- **User-dependent**, depend on the user's characteristics.
- **Query-dependent**, depend on the user's query.
- **Document-dependent**, depend on a single document.
- **Collection-dependent**, depend on information from the complete collection.

The characteristics of each signal will impact its computation.

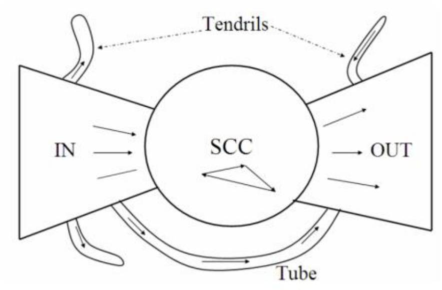
Examples of ranking signals include keyword relevance, page authority, backlink quality, content quality, user engagement metrics, page loading speed, mobile-friendliness, social signals, freshness of content, structured data markup, geographic relevance, and query-specific relevance.

- **5. What are the SCC, IN and OUT components in the view of the web as a bowtie?**

In the web's bowtie structure, **SCC (Strongly Connected Component)** represents the core interconnected region where pages can reach each other. **IN** components are pages linking into the SCC, and **OUT** components are pages linked out from the SCC, forming a bowtie-shaped topology.

A web surfer can pass from any page in IN to any page in SCC, by following hyperlinks. Likewise, a surfer can pass from page in SCC to any page in OUT.
Finally, the surfer can surf from any page in SCC to any other page in SCC.

However, it is not possible to pass from a page in SCC to any page in IN, or from a page in OUT to a page in SCC (or, consequently, IN).



► **Figure 19.4** The bowtie structure of the Web. Here we show one tube and three tendrils.

Completed by Sofia Merino Costa
up202300565

11. Link analysis (L9)

[Web ranking, anchor text, PageRank, hubs and authorities.]

Web Overview and Challenges (WWW):

- The web is unprecedented in many ways:
    - Unprecedented in scale (size and change);
    - Unprecedented in lack of central coordination;
    - Unprecedented in the diversity of users' backgrounds and needs.
- **Two types of challenges**:
    - **Data**: distribution, size, volatility, quality, unstructured, duplicates.
    - **Interaction**: user needs; relevance; diversity of users.

Web Search Challenges:

- Decentralization of content publication (the web biggest innovation) means that there is no central point that keeps track of what exists, what was changed, or deleted.
- What is the size of the web?
- Content is created massively and in diverse forms
    - Diversity of languages and dialects;
    - Diversity of formats, i.e. structure, colour, size, ...;
    - Quality, i.e. truths, lies, contradictions, ...;
    - Intent, i.e. legitimate, spam, search engine manipulation;

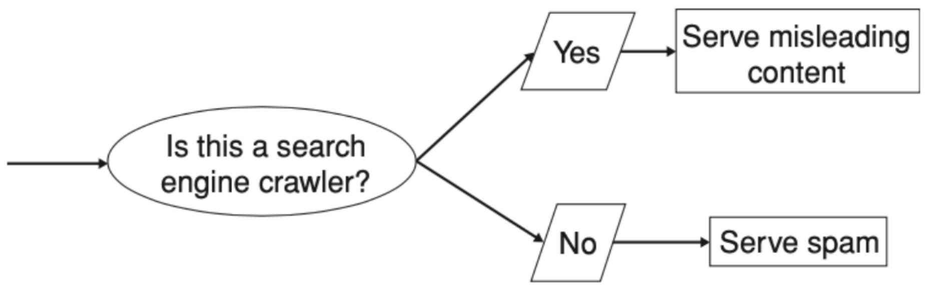
Web Characteristics:

- The Web can be modelled as a graph.
- Web pages point to, and are pointed by, other pages.
- **Links to other pages (out-links)** usually include an "anchor text". The number of **in-links to a page** is called the in-degree.
- Studies of web characteristics and dynamics is an area of research.
- Early research suggests that the directed graph connecting web pages has a bowtie shape.

Web Spam:

- There is a (high) commercial value associated with appearing on the top ranked results for a given search.
- Search engines must be resistant to manipulation attempts (high frequency!).
- For instance, a search engine whose scoring depends on the frequency of keywords, would be easy to manipulate by including numerous repetitions of selected keywords.
- This is called web spam, i.e. the manipulation of content on the web with the purpose of manipulating search engine rankings. Examples include cloaking, link farms, link spam, click spam, etc.
- Topics in the sub-area of adversarial information retrieval.

Cloacking:



User Characteristics:

- Web search users are not trained on how to write queries using the search operators offered by search engines.
- 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.
- Early research has shown that users' information needs can be grouped in three types:
    - Informational queries, i.e. seek general information about a topic. There is typically not a single source of relevant information. Users typically gather information from multiple web pages.
    - Navigational queries, i.e. seek the website or home page of a single entity. Users' expectations is to find a specific resource.

  - Transactional queries, i.e. preludes the user performing a transaction on the web, such as purchasing a product, downloading a file, or doing a reservation.

Signals for Web Ranking:

- Hundreds of signals are used by search engines to estimate quality.
- **Signals can be grouped by different dimensions:**
  - Query-independent signals (static).
  - Query-dependent signals (dynamic).
  - Document-based signals (content or structural), e.g. HTML.
  - Collection-based signals, e.g. Links.
  - User-based signals, e.g. Clicks.

Web Crawling:

- Crawling the Web, in a certain way, resembles watching the sky in a clear night: the star positions that we see reflects the state of the stars at different times.
- **Web crawling is the process by which pages are gathered from the web.**
- The **goal** is to discover as quickly and as efficiently as possible quality web pages.
- **Features a crawler must provide:**
  - robustness in face of problems, unexpected content, or traps,
  - politeness to web hosts.
    - A crawler cannot overload a web site with HTTP requests;
    - That implies that a crawler should wait a small delay between two requests to the same web site.
    - A crawler that is impolite may be banned by the hosting provider.
- **Features a crawler should provide:**
  - 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.
- **Three basic rules for web crawler operation:**
  - A crawler must identify itself as such and must not pretend to be a regular user.
  - A crawler must obey the robot's exclusion protocol (robots.txt).
  - A crawler must keep a low bandwidth usage in a given web site.

Robot Exclusion Protocol:

- The robot exclusion protocol involves three types of exclusion:
    - server-wide
    - page-wise exclusions
    - cache exclusions.
- **Server-wide exclusion** instructs about directories that should not be crawled.
    - These exclusions are defined via the robots.txt file located at the root.
- **Page-wise exclusion** is done by the inclusion of meta tags directly on in the pages.
    - Meta tags are part of the HTML standard.
- **Cache exclusion** is used to instruct search engines not to show the user a local cached copy of the page.

Applications of Web Crawling:

- A web crawler can be used to
    - create an index covering broad topics (general web search);
    - create an index covering specific topics (vertical web search);
    - archive content (archiving, backup);
    - analyse web sites for extracting aggregate statistics (characterization, analytics); keep copies or replicate web content (mirroring);
    - web site analysis.

Taxonomy of Crawling and Taxonomy of Web Pages... (Slide 25-26)

Web Crawling Process:

- The crawler starts with a set of seed pages,
    - these are downloaded, parsed, and scanned for new links.
- The links to pages that have not yet been downloaded are added to a central queue for download later (URL frontier).
- The crawler selects a new page for download and the processes is repeated:
    - until a stop criterion is met.
    - considering an established update / refresh policy.

Web Ranking:

- Ranking is the most complex and most important function of a search engine.
- **First challenge,** implement an evaluation process:
  - Devise an adequate process of evaluating the ranking in terms of relevance of results.
  - Such evaluation is necessary to fine tune the ranking function.
- **Second challenge**, identify quality content:
- Collect and combine evidence of quality from different signals.
- **Third challenge,** avoid, prevent and manage web spam.
- **Fourth challenge,** define a ranking function and compute it.

Ranking Signals:

- Signals can be organized in **different types**.
  - **Content signals**
    - related to the text itself.
    - consider HTML semantics (headings, sections, links).
  - **Structural signals**
    - related to the link structure of the web.
    - can be textual (e.g. anchor text), or related to the links (e.g. number).
  - **Usage signals**
    - related to the feedback provided by the users (e.g. clicks).
    - other usage signals include geographical location, technological context, temporal context.

Signals can also be distinguished according to other **dimensions**.

- **User-dependent**, depend on the user's characteristics.
- **Query-dependent**, depend on the user's query.
- **Document-dependent**, depend on a single document.
- **Collection-dependent**, depend on information from the complete collection.

Learning to Rank:

- Given dozens or hundreds of signals, how to combine them in a ranking function?

- **Traditional approaches** rely on manual selection of signals and respective weights, depending on domain knowledge.
- **Modern approaches** are based on machine learning techniques to:
    - Learn the ranking of the results.
    - Given a query Q, different training data can be used:
        - **pointwise**, a set or relevant pages for Q.
        - **pairwise**, a set of pairs of relevant pages indicating the ranking relation between the two pages.
        - **listwise**, a set of ordered relevant pages.

Link Analysis:

- Link Based **Signals**:
    - Links are one of the distinctive features of a collection of web documents.
    - Base assumption: the number of hyperlinks pointing to a page provides a measure of its popularity and quality.
    - Link-based ranking algorithms build on the assumption that an hyperlink from page A to page B represents an endorsement of page B, by the creator of page A.
    - Two classic algorithms: PageRank and HITS.
        - PageRank, Larry Page and Sergey Brin (1996)
        - HITS, Jon Kleinberg (1997)
- **Page Rank:**
    - The PageRank of a node is a **value between 0 and 1**.
    - It is a **query-independent** value computed offline that only depends on the structure of the web graph.
    - The algorithm models a random surfer who begins at a web page and, at each step, randomly chooses between the out-links to move to the next page. If the random surfer executes this "forever", he will visit some pages more frequently than others. The PageRank value of a page represents this probability.
    - $$PR(a) = \frac{q}{T} + (1-q)\sum_{i=1}^{n}\frac{PR(p_i)}{L(p_i)}$$

- **HITS (Hyperlink Induced Topic Search):**
    - **Query-dependent** algorithm.
    - Starts with the answer set (e.g. pages containing the keywords).
    - 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**.
    - $$H(p) = \sum_{u \in S \mid p \to u} A(u) , \qquad A(p) = \sum_{v \in S \mid v \to p} H(v)$$

Anchor Text as a Signal:

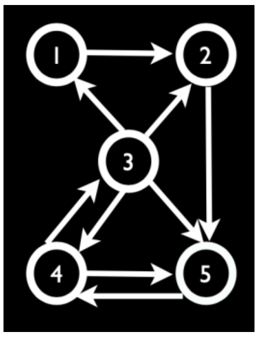
- **The text used in HTML anchors, i.e. links, is called anchor text.**
- <a href="http://www.fe.up.pt">FEUP</a>
- Represents a description from "others" about a given web page.
- The collection of all anchor texts can be explored with standard IR techniques and incorporated as an additional feature in an inverted index.
- Important feature for image search.

Questions:

- **1. What are in-links and out-links for a web page?**

- **2. How is anchor text used in web search?**

- **3. Calculate PageRank values for a set of linked documents.**



Completed by Sofia Merino Costa
up202300565

PR1 = PR3 / 4

PR2 = PR1 / 1 + PR3 / 4

PR3 = PR4 / 2

- **4. Calculate Hub and Authority values for a set of linked documents.**

Auth1 = (Hub3) / SUM (All Hub Votes)

Auth2 = (Hub1 + Hub3) / SUM (All Hub Votes)

Auth3 = (Hub4) / SUM (All Hub Votes)

Hub1 = (Auth2) / SUM (All Auth Votes)

Hub2 = (Auth5) / SUM (All Auth Votes)

Hub3 = (Auth1 + Auth2 + Auth4 + Auth5) / SUM (All Auth Votes)

12.Learning to Rank and Neural Information Retrieval (L10)

Learning to Rank:

- is a task to automatically construct a ranking model using training data, such that
- the model can sort new objects according to their degrees of
    - relevance, preference, or importance.

Ranking Problems:

- Many information retrieval problems are by nature ranking problems, such as:
    - document retrieval,
    - collaborative filtering,
    - key term extraction,
    - definition finding.
- Many different ranking scenarios:
    - Rank document purely according to their relevance with regards to the query.
    - Considering the relationships of similarity and diversity between documents (e.g. relational ranking).
    - Aggregate several candidates ranked lists (e.g. meta search, unified ranking).
    - Find to what degree a document property influences the ranking result.

Ranking in Information Retrieval:

- Conventional document ranking models:
    - **Query-dependent models:**
        - Boolean model, set based model (no degree of relevance)
        - Vector space model, using term weighting such as TF-IDF.
        - Probabilistic models, such as language models and BM25.
    - **Query-independent models:**
        - Link-based models for the web, such as PageRank.

Problems with LTR:

- Many of the existing ranking models contain parameters that need to be tuned. Different ranking models can be combined to create a new ranking.
- In addition, models perfectly tuned using the validation set do not necessary perform well on unseen queries.
- How to define and combine parameters and models to produce an effective ranking?

    - Machine learning methods have demonstrated their effectiveness in automatically tuning parameters combining multiple evidence.
    - **Learning to rank is the application of machine learning methods to information retrieval problems**.

Typical Learning to Rank Flow:

- The training data consists of:
    - n queries (q1, q2, ...),
    - their associated documents represented as features vectors (x1, x2, ...),
    - and the corresponding relevance judgements.
- A learning algorithm is employed to learn the ranking model h.
- The ranking model h is then evaluated with standard IR measures by producing ranked lists for the training data.

Learning to Rank Approaches:

- The learning to rank process can be modelled in different ways.
    - Pointwise approach
    - Pairwise approach
    - Listwise approach
- Different learning techniques can be employed, including SVM, Boosting, Neural Nets, etc.
- **Effectiveness from literature: Listwise > Pairwise > Pointwise**
- Different methods can be combined with successful results.

Pointwise Approach:

- **Input**: feature vectors for single documents, e.g. scores for query-document pairs.
- **Output**: relevance degree of each single document (e.g., relevant / non-relevant).
- **Methods**: OC SVM, McRank, Prank.
- Treat the problem as a regression problem, i.e. estimate a continuous variable (i.e. each document's score).
- Straightforward approach.
- Ambitious goal — produce document scores, but only rankings are needed.

Pairwise Approach:

- Pairwise classification problem, i.e. decide pairwise preferences for documents.
- **Input**: pairs of documents, both represented as feature vectors.
    - Manually annotated data, e.g. (q,d,d') — d is more relevant to q than d' (harder to get, high quality).
    - Log data, e.g. if, for query q, document d and d' are listed, and user clicked d and not d', then (q,d,d') can be inferred (easy, lower quality).
- **Output**: pairwise preferences (ranging from 1 to -1) between document pairs.
- The problem is treated as a classification problem, i.e. given a query and two documents, determine which is better.
- Methods: Ranking SVM, LambdaMART, LambdaRank

Listwise Approach:

- **Input**: entire group of documents associated with a query, i.e. **ranking lists**.
    - **Offline**: obtain relevance judgements (q,d,s), where s is a score indicating the relevance of document d to query q.
    - **Online**: present different rankings to users (or interleave lists) and observe (from logs) which users prefer. Output: full document predicted ranking for query.
- Methods: PermuRank, AdaRank, SoftRank.
- Problem is modelled in a more natural "IR" way but not directly adaptable to conventional machine learning techniques.
- **Best performing approaches**.

Neural information retrieval definition, concepts, and application to the retrieval process.

Neural Information Retrieval Concepts:

- **Neural Networks**: computational model consisting of layers of interconnected nodes that process and transform input data to produce an output.
    - **Shallow NN**: uses only one or two layers; used in "shallow learning" tasks.
    - **Deep NN:** uses multiple layers; used in "deep learning" tasks.
- **Deep Learning**: subset of machine learning that uses neural networks with multiple layers to learn complex patterns from data.

- **Dense Vectors**: numerical vector representations where most elements are non-zero; used to encode dense data. Sparse Vectors, numeral vector representations where most elements are zero; used to encode sparse data.
- **Embeddings**: low-dimensional numerical vector representations of objects (e.g., queries, text, images); able to capture relationships and similarities.
    - **Word Embeddings**: specific type of embedding used in NLP to represent words.
- **Transformers:** neural network architecture with high impact in NLP problems; efficiently captures relationships between words.
    - **Encoder**: component in neural networks that processes input data to an encoded representation (e.g., dense vector).
    - **Decoder**: component in neural networks that generates output based on the encoded representation.
- **BERT** (Bidirectional Encoder Representations from Transformers), pre-trained language model built on transformer architecture.
- **Neural IR:** application of neural networks in the context of information retrieval tasks.
- **Semantic Search:** application of search techniques that leverage on understanding the meaning (semantics) of user queries and documents to improve relevance estimation.
- **Dense Retrieval:** use of dense vector representations for document retrieval and ranking.
- **Language Model:** learned computational model representing text; used in many NLP tasks.
- **Large Language Model:** language models that use many parameters (i.e., millions or billions).

Neural Information Retrieval:

- Neural IR is the application of shallow or deep neural networks to IR tasks.
    - Neural models have been employed in many IR scenarios—including ad-hoc retrieval, recommender systems, multimedia search, and even conversational systems that generate answers in response to natural language questions.

- Unlike Learning to Rank approaches that train machine learning models over a set of hand-crafted features, neural models accept the raw text of a query and document as input.
- Neural networks are typically used in two different ways:
    - Learn the ranking functions combining the relevance signals to produce an ordering of documents.
    - Learn the abstract representations of documents and queries to capture their relevance information.

Data in Neural Information Retrieval:

- Neural models for IR use vector representations of text, which usually contain many parameters that need to be tuned.
- ML models with large set of parameters typically benefit from large quantity of training data.
- Learning suitable representations of text demands large-scale datasets for training.
- Therefore, unlike classical IR models, these neural approaches tend to be data hungry, with performance that improves with more training data.

Document Ranking (check images):

- Document ranking comprises three steps (recall “IR models” topic):
    - Generate a representation of the query that specified the information need.
    - Generate a representation of the document.
    - Match the query and the document representations to estimate their mutual relevance.
- Neural approaches can influence one or more of these three steps.
- Unlike traditional learning to rank models, neural architectures depend less on manual feature engineering on more on automatically detecting regularities in matching patterns.
- Many neural IR models depend on learning low-dimensional vector representations (or embeddings) of query and document text.
- And then use them with traditional IR models in conjunction with simple similarity metrics (e.g., cosine similarity).

Word Embeddings:

- In the 1950s, many linguists formulated the distributional hypothesis: words that occur in the same contexts tend to have similar meanings.
- According to this hypothesis, the meaning of words can be inferred by their usage together with other words in existing texts.
- “You shall know a word by the company it keeps.”, John Firth (1957)
- In recent years, the distributional hypothesis has been applied to create semantic
- understandings of terms and term sequences through word embeddings.
- A word embedding is a numerical vector that represents the semantic meaning of a given term sequence.
- Word embeddings allow for the representation of words:
    - as low-dimensional
    - numerical dense vectors
    - that are learnt from data,
    - resulting in a projection in a new space
    - where similarities and relationships between the original words are kept.

Transformers:

- Static word embeddings map words with multiple senses into an average or most common-sense representation based on the training data used to compute the vectors.
    - The vector of a word does not change with the other words used in a sentence around it.
- A transformers is a neural network designed to explicitly take into account the context of arbitrary long sequences of text.
    - Transformer compute contextualized word embeddings, where the representation of each input token is conditioned by the whole input text.
    - The most popular contextualized word embeddings are learned with deep neural networks such as the Bidirectional Encoder Representations from Transformers (BERT).
- With fine-tuning, the parameters of a pre-trained language model can be updated for the domain data and target task.

    - Pre-training typically requires a huge general-purpose training corpus, and long and expensive computation resources.
    - On the other side, fine-tuning requires a small domain-specific corpus focused on the downstream task, affordable computational resources and few hours or days of additional training.

From "exact match" to "soft match":

- Key point: neural methods replace exact matching with soft matching.
- With traditional methods, soft matching is possible:
    - Stemming handles singular/plural and different tenses.
    - Synonyms expand soft match possibilities.
    - Query expansion techniques can add context to the query by adding new terms.
- With neural methods, soft match is central:
    - Word embeddings capture semantic similarities, allowing nuanced understanding of related terms.

Neural IR Summary:

- Deep learning methods greatly improve soft matching in document ranking.
- Pre-trained large language models, avoid the need for large amounts of training data to implement effective document retrieval solutions.
- However, a search system needs to support both soft and hard matching.
- Retrieval architectures need to combine traditional methods with neural IR methods.
    - E.g., first-stage neural selection (i.e. "semantic expansion"), followed by a standard retrieval stage over the selected documents.

Vector Search Systems:

- Also called "vector databases".
- In dense retrieval systems, document embeddings are pre-computed, thus the need for storing and searching through these document embeddings.
- Given a query, represented as a dense vector, instead of computing the distance between this vector and all document dense vectors (too expensive), the strategy is to use approximate nearest neighbour search.
- Solr (since version 9) includes support for Dense Vector Search.

Dense Vector Search in Apache Solr:

- Solr's Dense Vector Search adds support for indexing and searching dense vectors.
- Uses deep learning to produce vector representations for queries and documents.
- Dense Vector Representation:
  - Traditional tokenized inverted index vs. dense vector representation.
  - Distils approximate semantic meaning into a fixed number of dimensions.
  - Lower dimensionality compared to sparse vectors, offering efficiency.
- Dense Retrieval:
  - Explores the strategy for efficient retrieval using navigable small-world graph.
  - Provides approximate nearest neighbour search for high-dimensional vectors.

Dense Vector Indexing Configuration:

- A new field type is added — DenseVectorField:
  - Field type designed for dense vector search.
  - Indexing and searching dense vectors of float elements.
- Example configuration in schema using vectorDimension and similarityFunction.
- Advanced Configuration:
  - Configuring advanced parameters like knnAlgorithm, vectorEncoding, hnswMaxConnections, and hnswBeamWidth.
  - Customizing the codec format and hyper-parameters of the HNSW algorithm.

Semantic Search Tutorial:

- Indexing:
  - Uses Hugging Face sentence transformers for document encoding.
  - Adds a new field for storing document vectors.
- Retrieval:
  - Encode query using same model.
  - Uses Solr's k-nearest neighbors query parser to retrieve similar documents.

Questions:

- **1. What is Learning to Rank?**

[described above]

- **2. Which are the main approaches in LTR? How do they differ in terms of input and output data?**

[point, pair and listwise approaches]

- **3. What is Neural Information Retrieval?**

[described above]

- **4. How can neural models be used in the retrieval process?**

- **5. What are document embeddings?**

- **6. What is semantic search?**

[application of search techniques that leverage on understanding the meaning (semantics) of user queries and documents to improve relevance estimation.]

- **7. What is the difference between Learning to Rank and Neural Information Retrieval?**

[Learning to Rank focuses on **developing algorithms that learn to order or rank a set of items based on their relevance to a query**. On the other hand, Neural Information Retrieval involves the **application of neural network models, particularly deep learning techniques, to enhance various aspects of the information retrieval process**, including document representation, query understanding, and relevance ranking.]

13.Query processing (L11)

Query Processing:

- Once the necessary data structures are in place, we need to efficiently use them to obtain the search results in response to a user query.
- Two main query processing techniques:
    - **Document-at-a-time (Slide 9):** 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 (Slide 10):** 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.
- In both approaches, optimization techniques can significantly reduce the time required.

Optimization Techniques:

- Two classes of optimization techniques for query processing:
    - Read less data from the index.
    - Process fewer documents.
- When using complex feature functions, focusing on scoring fewer documents is the main concert.
- With simple feature functions, performance gains come from ignoring as much of the inverted list data as possible.
- Early termination of query processing:
    - Ignore high-frequency word lists in term-at-a-time, common terms have long postings lists, thus high processing costs.
    - Ignore documents at end of lists in document-at-a-time, when documents are ordered by some quality metric.
- Order postings in inverted indexes
    - Order inverted lists by quality metric (e.g. number of IN links in web IR).
- Caching
    - Cache popular query results

Skip Pointers:

- Skip pointers are used to speed-up inverted index scans.
    - i.e., reading less data from inverted lists.
- In doc-at-a-time, looking for documents in postings benefits with skip pointers, i.e. more quickly get to the document being scored.
- In term-at-a-time, skip lists can be used in the term accumulators, and in conjunction with 'conjunctive processing' to skip over common terms.

Conjunctive Processing:

- Base assumption: only return documents that contain all query terms.
- This is the default in web search engines and the default users' expectations.
- Conjunctive processing works best when one of the terms is rare.
    - [wine baga], since 'baga' is rarer, we can skip most of the inverted list for 'wine'.
- Can be employed for both document-at-a-time or query-at-a-time.
- In short queries, benefits both efficiency and effectiveness.
- Long queries, e.g. phrase searches, are not good candidates for conjunctive processing.

Relevance Feedback:

- Exact matches aren't the only way to obtain relevant results in search systems.
- The vocabulary mismatch between the user and the collection contribute to this problem. Also, the fact that synonyms exist.
- For example, a search for [aircraft] should also include results for [airplane].
- This can be addressed by manually refining the query.
- On the system side, this can be tackled with different techniques, broadly grouped in:
    - Global methods, expand or reformulate the query terms independently of the query or the results returned from it, e.g. using thesaurus, and using spelling correction.
    - Local methods, adjust a query relative to the documents that initially appear to match the query, e.g. relevance feedback, and pseudo-relevance feedback.
- The idea of relevance feedback is to involve the user in the process to improve the final result set by considering user feedback about the initial set of results.
- **Basic procedure** (one or more iterations are possible):

    - The user issues a (short, simple) query.
    - The system returns an initial set of retrieval results.
    - The user marks some returned documents as relevant or non-relevant.
    - The system computes a better representation of the information need based on the user feedback.
    - The system displays a revised set of retrieved results.
- **Relevance feedback explores the idea** that it may be difficult to formulate a good query when you don't know the collection, but it is easy to judge particular documents.
- Relevance feedback can be effective in tracking a user's evolving information need, i.e. seeing some documents may lead users to refine their understanding of the original information need.
- Image search is a good example of relevance feedback, a task where it may be difficult to express a query in words but is easy to determine if an image is relevant.

Rocchio Algorithm:

- The Rocchio algorithm is the classic algorithm for implementing relevance feedback.
- It models a way of incorporating relevance feedback information into the vector space model.
- It is 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.

Rocchio Algorithm Performance:

- Relevance feedback can improve both recall and precision.
- In practice it has been shown to be most useful for increasing recall — partially an effect of the use case: when a user wants high recall (i.e. see all relevant documents), they are expected to take the time to review results and iterate over the search.

- Positive feedback is more valuable than negative feedback. Many systems only allow for positive feedback.

Relevance Feedback Limitations:

- Cases where relevance feedback might be insufficient:
    - **Misspellings**, if the user spells a term in a different way to the way it is spelled in the document collection, relevance feedback is unlikely to be effective.
    - **Cross-language retrieval,** documents in another language are not near (in a vector space) of other topic related documents, rather documents in the same language are closer.
    - **Vocabulary mismatch**, in this cases the initial query mostly likely will fail and thus relevant feedback won't be effective.
- Users are often reluctant to provide explicit feedback. Users expect single interactions in search and the concept of relevance feedback is hard to explain to the average user.
- Additionally, it is often harder to understand why a particular document was retrieved after relevance feedback was applied.

Pseudo Relevance Feedback:

- Pseudo relevance feedback provides a method for automatic local analysis.
- It automates the manual part so that a relevance feedback algorithm is applied without extended user interaction.
- The method is applied to normal retrieval — assume that the top k ranked documents are relevant and apply a relevance feedback algorithm under this assumption.

Implicit Relevance Feedback:

- The use of indirect sources of evidence rather than explicit feedback on relevance.
- Implicit feedback is less reliable than explicit feedback, but is more useful than pseudo relevance feedback, which contains no evidence of user judgements.
- Clickstreams are one of the main examples of indirect relevance information — clicks on links are assumed to indicate that the page was likely relevant for the query.

Query Expansion:

- With query expansion, user give additional input on query words or phrases to suggest additional terms. Users opt to use one of alternative query suggestions.
- How to generate alternative question expansions for the user:
    - Use synonyms and related words from a global thesaurus.
        - Use a controlled vocabulary to build a thesaurus;
        - Use a manual thesaurus built by editors;
        - Automatically derive thesaurus, e.g. use text statistics;
        - Use query log mining to find related expansions (global, contextual, user-based);

Questions:

- **1. Describe and distinguish between the two query processing techniques — document-at-a-time and term-at-a-time.**

[described above]

- **2. In what contexts is query transformation / expansion advantageous?**

[Query transformation/expansion is advantageous in contexts where enhancing the original user query can improve information retrieval effectiveness, such as dealing with ambiguity, synonymy, or capturing diverse aspects of a user's information need.]

- **3. What techniques can be used to apply transformations / expansions to user queries?**

- **4. Identify and describe query expansions techniques, such as relevance feedback or pseudo-relevance feedback.**

[described above]

14.Entity-oriented search (L12)

Knowledge Bases and Entities:

- A primary component enabling these advanced search services is the availability of large- scale structured knowledge repositories, called knowledge bases.
- Knowledge bases organize information around specify things or objects, called "entities".
- An **entity** is a uniquely identifiable object or thing, characterized by its name(s), type(s), attributes, and relationships to other entities.
- **Common types of entities include** people, organizations, products, locations, and events.

Properties of Entities:

- **Unique identifier**: entities need to be uniquely identifiable. There must be a one-to-one correspondence between each entity identifier and the object it represents. Examples include: username, email address, URI, Wikipedia page ID, etc.
- **Name**(s): entities are known and referred to by their name. Unlike IDs, names do not uniquely identify entities; multiple entities may share the same name; and the same entity may be known by more than a single name.
- **Type**(s): entities may be categorized into multiple entities types. Types work as semantic categories that group together entities with similar properties. The set of possible entity type is often organized in a hierarchical structure, e.g. "scientist" is a subtype of "person".
- **Attributes**: the characteristics or features of an entity are described by a set of attributes. Different entities have different sets of attributes. Some of the characteristics may be entities themselves, in this we cases they are not treated as attributes but as relationships, e.g. "place of birth" links to a location entity. Attributes always have literal values and optionally may also include data type information.
- **Relationships**: describe how two entities are associated to each other. Relationships may also be seen as "typed links" between entities.

Representing Properties of Entities:

- Information about entities can be represented and stored in semi-structured or in structured form.

    - Wikipedia is an example of a knowledge repository that organizes information about entities and their attributes and relationships in a semi-structured form.
    - To adopt a structured form, a knowledge representation model is necessary. The Resource Description Framework (RDF) is a standard way to describe entities, where an entity is represented as a set of RDF statements.
- A knowledge base (KB) is a structured knowledge repository that contains a set of facts (assertions) about entities.
- Entities in a knowledge base may be seen as nodes in a graph, with relationships between them as edges. Thus, knowledge bases are also referred to as knowledge graphs.

Semantic Web:

- The Semantic Web is a term coined be Tim Berners-Lee in 2001 referring to an envisioned extension to the original web.
- While the original web is a medium of documents for people — a Web of Documents.
- The semantic web is meant to be a medium for machines (intelligent agents) that have access to data and rules for reasoning about the data — a Web of Data.
- Semantic Web technologies include URI, XML, RDF, various serializations of RDF (Turtle, N- Triples, RDFa, etc), SPARQL, OWL, etc — commonly referred to as the Semantic Web Stack.
- This vision is yet (?) to come true, however it has contributed to the development and publication of structured data in an unprecedented scale.

Entity-Oriented Search:

- **Entity-Oriented Search** is the search paradigm of organizing and accessing information centred around entities, and their attributed and relationships.
- **From a user perspective**, entities are natural units for organizing information. Allowing users to interact with specific entities offers a richer and more effective user experience than what is provided by conventional document-based retrieval systems.
- **From a machine perspective**, entities allow for a better understanding of search queries, of document content, and even of users. Entities enable search engines to be more effective.

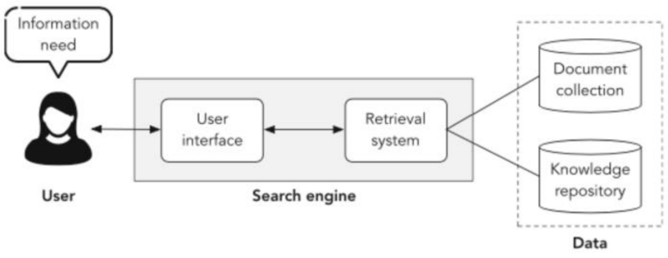


Fig. 1.3 Architecture of an entity-oriented search system

Users and Information Needs:

- Users articulate their information needs in many different ways.
- **Keyword queries:** also known as "free text queries", have become the most common approach with the advent of the web. Keyword queries are easy to formulate but imprecise.
- **Structured queries:** structured data sources are traditionally queried using formal query languages (e.g., SQL, SPARQL). These queries are very precise and usually intended for expert users with well-defined and precise information needs.
- **Keyword++ queries:** correspond to keyword queries that are complemented with additional structural elements, e.g. [retrieval site:fe.up.pt].
- **Natural language queries:** are formulated using natural language, the same way one would express it in a conversation, e.g. ["What is the capital of Nepal?"].
- **Zero-queries:** the traditional information access is reactive — the search system responds to a user-initiated query. Proactive systems, anticipate the user's information need, without requiring the user to issue a query. In practice, the context of the user is used to infer information needs.

Data:

- Data can be grouped in three types.
  - **Unstructured data,** which can be found in vast quantities in a variety of forms: web pages, spreadsheets, emails, tweets, etc. All these may be treated as textual documents, i.e. a sequence of words.
  - **Semi-structured data,** is characterized by the lack of rigid, formal structure. Typically, it contains tags or other markup to separate textual content from semantic elements, e.g. HTML data.

- **Structured data,** adheres to a predefined (fixed) schema and is typically organized in a tabular format (e.g., relational databases). The schema defines how the data is organized and imposes constrains to ensure consistency.

Tasks in Entity-Oriented Search:

- **Entity Retrieval — entities as the unit of retrieval,** between 40% to 70% of the queries in web search mention or target specific entities. Such queries are better answered by returning a ranked list of entities, as opposed to a list of documents. Challenges involved:
  - 1) how to represent information needs;
  - 2) how to represent entities;
  - 3) how to match those representations.
- **Entity Linking — entities for knowledge representation,** recognizing mentions of entities in text and associating these mentions with the corresponding entities in knowledge bases.
- **Entities for an enhanced user experience,** besides providing meaningful retrieval as information organization units, entities can improve the user experience throughout the entire search process.

Data Sources:

Data Types:

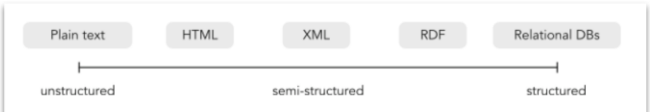


Fig. 2.1 The data spectrum

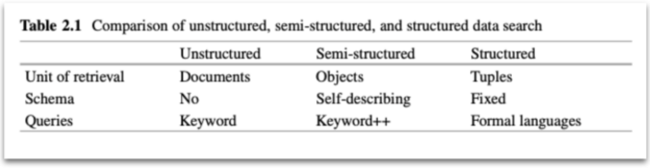


Table 2.1 Comparison of unstructured, semi-structured, and structured data search

	Unstructured	Semi-structured	Structured
Unit of retrieval	Documents	Objects	Tuples
Schema	No	Self-describing	Fixed
Queries	Keyword	Keyword++	Formal languages

Knowledge Bases:

- A **knowledge base** is comprised of a large set of assertions about the world.
- When the emphasis is on the relationships between the entities, the term knowledge graph is commonly used.
- **Resource Description Framework** (RDF) is a language designed to describe "things", which are referred to as resources.

- Each resource is assigned a **<u>Uniform Resource Identifier</u>** (URI), making it uniquely and globally identifiable. Each RDF statement is a triple, consisting of subject, predicate, and object components
  - **Subject**, always a URI, denoting a resource;
  - **Predicate**, always a URI, corresponding to a relationship or property of the subject resource;
  - **Object**, either a URI (referring to another resource) or a literal.

Entity Ranking:

Ad Hoc Entity Retrieval Task:

- Ad hoc entity retrieval is the task of answering queries with a ranked list of entities.
- "Ad hoc" refers to the standard form of retrieval in which the user, motivated by an ad hoc (extemporary) information need, initiated the search process by formulating and issuing a query.

Main Strategy:

- Model the "entity retrieval" problem as a "document retrieval" problem and take advantage of all the existing body of work.
- Create an entity description, or "profile document", for each entity in the catalog, containing all the existing "knowledge" about that entity, based on the available data.
- Once these entity descriptions are created, they can be indexed and ranked using existing document retrieval algorithms.
- Involves two steps, discussed in the next slides:
  - Constructing the profile documents;
  - Ranking the profile documents.

Constructing Entity Descriptions:

- First step is to create a "profile document" with all the information we have about that entity.
- This will serve as the textual representation of the given entity, the "entity description".

- Entity descriptions can be assembled by considering the textual content, from a document collection, in which the entities occur.
- In some cases, such descriptions may already be easily available — e.g., Wikipedia page of an entity. Also, there is a lot of information about entities organized and stored in knowledge bases.
- Three possible data sources: unstructured, semi-structured and structured.
- The goal is to estimate a term count associated with each entity.

Entity Components:

- The estimative of the number of times a term is associated with an entity can be used to compute standard components, in an analogous way to the document model.
- **Entity length** is the total number of terms in the entity description.
- **Term frequency (TF)** is the normalized term count (by length) in the entity description.
- **Entity frequency (EF)** is the number of entities in which the term occurs.
- **Inverse entity frequency (IEF)** is the log normalized ratio between the total number of entities in the catalog, and the entity frequency.

Entity Representations from Unstructured Data:

- This is the scenario where we want to find entities in arbitrary document collections.
- An approach is to have documents annotated with the entities that are references in them and use these documents as a proxy to connect terms and entities.
- These annotations may be provided by human editors or be automated (e.g., entity linking).
- In this scenario, the estimated entity term count is obtained by considering each document and adding the number of co-occurrences between a term and an entity, weighted by the strength of the association between the entity and the document.

Entity Representations from Semi-Structured Data:

- A significant portion of the web content is already organized around entities, e.g., Wikipedia page about a person, IMDb page about a movie or an actor.

- The standard way of incorporating the internal document structure into the retrieval model is by using document fields, where document segments correspond to fields — e.g., title, summary, introduction, etc.
- The standard statistics are now computed at field-level, e.g. the number of times a term appears in a field of the description of an entity.
- A special "catch-all" field is introduced to
    - (1) quickly filter entities in a first-pass retrieval;
    - (2) since entity description fields are often sparse, improve ranking by combining field-level scores with entity-level (the catch-all field) scores.

Entity Representations from Structured Data:

- There is a growing number of large general-purpose publicly available knowledge bases.
- In knowledge bases information is organized using the RDF data model, specifically
    - each entity is uniquely identified by its URI (Uniform Resource Identifier);
    - its properties are described in the form of subject-predicate-object (SPO) triples.
- To represent an entity from a knowledge base, we consider the immediate vicinity of the entity node — i.e. the SPO tripes where the entity appears either as subject or object.
- From the perspective of constructing entity representations, this is conceptually no different from having data stored in relational databases, where the same information is available through fields and foreign-key relationships.

Entity Representations from Triples:

- The number of potential fields is huge (in the 1000s), thus we need a different strategy.
- A common solution is predicate folding — grouping predicates together into a small set of predefined categories, so that we have a handful go fields.
- Predicates may be grouped based on their type or importance.

- Common entity fields: name, name variants, attributed, types, outgoing relations, incoming relations, top predicates, catch-all.

Commonly Used Fields:

- **Name** contains the name(s) of the entity.
    - The two main predicates mapped to this field are <foaf:name> and <rdfs:label>.
    - One might follow a simple heuristic and additionally consider all predicates ending with "name", "label", or "title".
- **Name variants (aliases)** may be aggregated in a separate field.
    - In DBpedia, such variants may be collected via Wikipedia redirects (via <dbo:wikiPageRedi- rects>) and disambiguations (using <dbo:wikiPageDisambiguates>).
- **Attributes** includes all objects with literal values, except the ones already included in the name field.
    - In some cases, the name of the predicate may also be included along with the value, e.g., "founding date 1964" (vs. just the value part, "1964").
- **Types** holds all types (categories, classes, etc.) to which the entity is assigned.
    - Commonly, <rdf:type> is used for types;
    - In DBpedia, <dct:subject> is used for assigning Wikipedia categories, which may also be considered as entity types.
- **Outgoing relations** contains all URI objects, i.e., names of entities (or resources in general) that the subject entity links to.
    - If the types or name variants fields are used, then those predicates are excluded;
    - Values might be prefixed with the predicate name, e.g., "spouse Michelle Obama".
- **Incoming relations** is made up of subject URIs from all SPO triples where the entity appears as object.
- **Top predicates** may be considered as individual fields
    - e.g., include the top-1000 most frequent DBpedia predicates as fields.
- **Catch-all** (or content) is a field that amasses all textual content related to the entity.

Triples to Text:

- How to represent triples as text?
- Triple objects values are either URIs (links) or literals.
- Literals can be treated as regular text.
- URIs need to be converted to human-readable
  - Some are user-friendly: https://dbpedia.org/page/Porto
  - Others are not: http://www.wikidata.org/entity/Q36433
- URI resolution is the process of finding a human-readable name/label for a URI.

URI Resolution:

- The specific predicate that holds the name of a resource depends on the RDF vocabulary used.
- Commonly, <foaf:name> or <rdfs:label> are used.
- Given a SPO triple, for example:
  - <dbr:Audi_A4> <rdf:type> <dbo:MeanOfTransportation>
- The corresponding resources' name is contained in the object element of this triple:
  - <dbo:MeanOfTransportation> <rdfs:label> “mean of transportation"

Ranking Entities:

- With the term-based entity representations, we now need to rank entities with respect to their relevance to a search query.
- This can be viewed as a the problem of assigning a score to each entity in the catalog.
- The retrieval models from document ranking — e.g. vector space model, language models, probabilistic models — are used for entity scoring by replacing the document with entity in the equations.

Entity Ranking Evaluation

- Evaluation of ad hoc entity retrieval is analogous to that of ad hoc document retrieval.
- Given a ranked list of items, the relevance of each item is judged with respect to the underlying information need.
  - Set-based measures, e.g. precision, recall, precision at rank cutoff k (P@10, P@20).

    - Rank-based measures, e.g. average precision (AvP), mean average precision (MAP).
- The use of text collections is the de facto standard standard for evaluation in IR. Some relevant collections that can be used for entity retrieval are: INEX Entity Ranking (XER), TREC Entity, Semantic Search Challenge, INEX Linked Data.

Entity Ranking Summary:

- In this section we discussed approaches for ranking entities in various datasets, from unstructured documents, to structured knowledge bases.
- Most of the effort in this area has been in constructing term-based representations of entities, which can be ranked using traditional document retrieval techniques.
- Unstructured entity representations with a bag-of-words retrieval models usually provide solid performance and a good starting point.
- Other approaches exist, most notably the use of specific characteristics of entities, such as relationships.

Entity Linking:

- Entity linking is the task of recognizing entity mentions in text and linking them to the corresponding entries in a knowledge base.

Anatomy of an Entity Linking System:

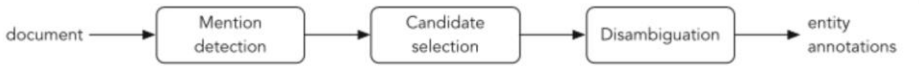


Fig. 5.2 Entity linking pipeline

- **Mention detection**, identification of text snippets that can potentially be linked to entities.
- **Candidate selection,** generating a set of candidate entities for each mention.
- **Disambiguation**, selecting a single entity (or none) for each mention, based on the context.

Common Approach for Mention Detection:

- Build a dictionary of the entity surface form, i.e. entities with all name's variants.
- Check all document n-grams against the dictionary
- Filter out undesired entities

Questions:

- **1. What is entity-oriented search? What is necessary to implement it?**

- **2. Describe the challenges and techniques associated with… building entity descriptions, entity ranking, entity linking.**

- **3. Describe the data sources typically required for entity oriented search and its characteristics.**

15. Search user interfaces (L11 & L13)

Search user interface design for search services

design principles and heuristics,

application examples.

A Framework for Thinking About Search Features:

- Max L. Wilson proposes a framework to help on the discussion of search user interfaces, elements and features.
- It divides the elements of a search user interface in 4 main groups:
  - **Input** - features that allow the searcher to express what they are looking for.
  - **Control** - features that help searchers to modify, refine, restrict, or expand their input.
  - **Informational** - features that provide results or information about results.
  - **Personalization** - features that relate specifically to searchers and their previous interactions

Related Disciplines:

- Six disciplines, or factors, contribute to the design of a search user interface.
  1. User Experience
  2. Graphic Design
  3. Information Retrieval
  4. Library & Information Science
  5. Info-Seeking
  6. Human-Computer Interaction

Search UI Design Impact Factors:

- Organization and properties of the data sources, e.g. available metadata.
- IR algorithms adopted, considering both speed and effectiveness.
- Graphic design, aesthetics impact how people judge the trustworthiness of a website. Simple use of colour and other simple visual cues can have significant impact on SUI success.
- How people search for information — field of Information Seeking Behaviour — also impacts on search user interfaces design.

UI Design Principles:

- General users interface design guidelines from the HCI and UX communities.
  1. **Visibility**, keep the user informed about what is happening at any given time. Example: keep the current search terms in the query box.
  2. **Language**, adopt language that the user can understand. Example: instead of 'query' use the term 'search', or instead of 'query expansion' use 'related searches'.
  3. **Control and freedom**, do not block users in a hole or fixed pathway, instead provide mechanisms for users to recover from them. Example: highlight spelling errors or mistakes but do not force them on users.
  4. **Consistency**, adopt a consistent design that follows the same conventions. Example: always use 'search' and not 'query' or 'keyword search' in other places.
  5. **Error prevention**, make it hard to do unproductive things, i.e. avoid the need to undo actions.
  6. **Support recognition**, help users not have to remember what they have done or need to do. Example: provide related searchers, keep the query in the search box.
  7. **Flexibility and efficiency**, provide features and shortcuts for experiences users to be more productive and efficient. Example: let users navigate search results with keyboard shortcuts.
  8. **Aesthetics and minimalism**, keep design simple and minimalist; make sensible use of white space.
  9. **Clear error messages**, provide informative and useful error messages. Example: not only state the lack of results but provide alternative searches.
  10. **Help and documentation**, provide help as documentation, FAQs, and examples.

Evaluating Search User Interfaces:

- Three approaches: IR style, empirical user studies, analytical approaches.
- IR style
  - Traditionally, IR systems are evaluated within a TREC-style environment, based on datasets, specific tasks, and known 'best results' for each task (calculated by human experts).

    - Given the success of the TREC approach, an interactive track was created to take search interaction into account. But success was limited. Simple evaluation measures like precision and recall were not sufficient.
- Other approaches are empirical and analytical.

Empirical User Studies:

- HCI user studies methods instead focus on how well a system, including the SUI, allows searchers to complete a task.
- It became common to evaluate search systems by creating user studies using tasks that are specifically oriented towards search.
- Empirical methods are about observing and recording actual user performance.
- Common measurements used in user studies: number of searches, number of terms per search, number of results visited, search times, task accuracy, etc. Qualitative methods such as interviews and observations are also possible.
- Designing and conducting user studies is hard. Results can be impacted by many factors such as the motivation of participants, software bugs, and even small user experience differences, e.g. slight differences in colors.

Analytical Approaches:

- In analytical approaches, typically low-cost or "discount" inspection methods are used to allow evaluators to assess a design and make well informed predictions about SUI.
- There are many analytics methods for UI and UX — e.g. heuristic evaluation, cognitive walkthrough —, but fewer specifically designed for SUI.
- Analytical approaches can leverage the wealth of models and experience previously developed by experts.
- Analytics methods can only make estimates about how suitable a design is, before a formal evaluation.

Choosing an Evaluation Approach:

- The HCI community has developed the DECIDE process to help on this decision:
    - D - **Determine** the goals of the evaluation;
    - E - **Explore** the specific questions to be answered;
    - C - **Choose** an evaluation paradigm, such as systematic IR or empirical user studies;

  - I - **Identify** practical issues in performing such an evaluation;
  - D - **Decide** how to deal with any ethical issues;
  - E - **Evaluate**, Interpret, and present the data.

Summary:

- Multidisciplinary, addressing these challenges requires knowledge and expertise from multiple areas, e.g. information science, visual design, human-computer interaction.
- Different types of features, search user interfaces features can be grouped in different types — input features; control features; informational features; personalization features.
- **Heuristics**, existing guidelines and best practices can help on the design of successful search user interfaces.
- User studies, evaluation of search user interfaces is done by observing users conducting specific tasks.

Questions:

- **1. Identify and describe user interface techniques and elements that can be used to improve user experience in using search systems.**

Input - features that allow the searcher to express what they are looking for.

Control - features that help searchers to modify, refine, restrict, or expand their input.

Informational - features that provide results or information about results.

Personalization - features that relate specifically to searchers and their previous interactions

- **2. Describe how user interaction innovations and experiments can be evaluated.**

Three approaches: IR style, empirical user studies, analytical approaches, HCI Community's DECIDE Process.

- **3. What are design principles and heuristics?**

Design Principles: ...

Heuristics: existing guidelines and best practices can help on the design of successful search user interfaces.

16. Solr (L6)

Apache Solr:

- Pronounced "solar"
- Is an open-source text centric search platform written in Java.
- Uses Apache Lucene for full-text indexing.
- Interaction is based on a REST-like HTTP XML/JSON API.

Lucene and Solr:

- **Apache Lucene** is a search library written in Java that provides the fundamental building blocks for implementing indexing and search capabilities.
  - **Key features**: indexing, searching, ranking.
  - **Use cases**: directly manage and embed the indexing and integrate search within software's logic.
- **Apache Solr** is a search platform that uses Lucene and is prepared to be deployed and used as a standalone server in large-scale scenarios with large volumes of data.
  - **Key features**: scalability, REST API, user admin interface, search features (faceting, highlighting, autocomplete).
  - **Use cases**: deploy scalable, ready-to-use, search platform that works as a standalone server and is integrated with other services through an API.

Solr Installation:

- You can start a Solr server with:
  - docker run --name my_solr -d -p 8983:8983 solr:9
  - --name, defines a name for the container
  - -d, starts the container in detached mode to free up the terminal
  - -p, maps Solr's default port from the container to the host machine.
  - Different versions can be selected with 'solr:version'. Omitting the version defaults to the latest.
- Head to http://localhost:8983 to access Solr Admin user interface.

Solr Key Concepts:

- Solr is a search framework that indexes data and then enables retrieval of that data. The basic information unit in Solr is a **document** composed of **fields**.
- Document fields can be of specific **types** (date, number, currency, text, uuid, ...).

- **Textual fields** go through a pipeline of analyzers, tokenizers, and filters.
  - **Analyzers**, receive a textual field as input and generates a token stream.
  - **Tokenizers**, receive a character stream and produce a sequence of token objects.
  - **Filters**, examine tokens and transform them (keep, discard, create, modify).
- These pipelines are applied per field in the indexing process and to the query in the querying process.
- Definitions for field types and field configurations are defined in the **schema file**.
- A Solr instance can have multiple cores (or indexes).
- A **Solr core** stores information for the indexed documents.
- Solr supports a REST-like HTTP XML/JSON **API** for both indexing and querying.
- **Queries**, just like documents being indexed, also go through a pipeline of analyzers, tokenizer, and filters.
  - But the pipelines for documents and queries can be different.
- **Query parsers** convert a search string into a Lucene query and find the matching documents.
  - Different query parsers exist to support different search requirements.
  - E.g. the standard query parser, eDisMax (Extended DisMax) query parser.

Index Documents (Schemaless):

- Documents can be posted (indexed) into a newly created core without further definitions. This is referred to as “schemaless mode”.
- The are two main ways to post document to a core, using the API or the post tool.
- **Using the post tool requires that the data is available to the container.**
  - One option is to map a local folder to a container folder, as shown in the tutorial.
  - Another option is to use “docker cp ...” to copy the data into the container.
- Using the API requires using curl to submit an **HTTP POST command**.
  - curl -X POST -H 'Content-type:application/json' \ --data-binary "@./pt-news.json" \ http://localhost:8983/solr/ptnews/update\?commit\=true

Performing Queries:

- Queries can be performed **directly using Solr Admin** > Query
  - Example: q: “content:portugal”
- Or **submitting an HTTP GET JSON request** using curl (or other tool for HTTP requests).
  - curl http://localhost:8983/solr/ptnews/query -d '{ "query": "*:*"}'

    - curl http://localhost:8983/solr/ptnews/query -d '{"query": "content:portugal"}'
- Queries can also be executed **using direct HTTP requests** (i.e. no JSON):
    - curl 'http://localhost:8983/solr/ptnews/select?q=content:portugal'
- Example on how to **define the fields to retrieve**:
    - curl 'http://localhost:8983/solr/ptnews/select?q=*&fl=id,title'
- Limitations of the **lack of schema**. Compare:
    - curl http://localhost:8983/solr/ptnews/query -d '{ "query": "content:orcamento"}'
    - curl http://localhost:8983/solr/ptnews/query -d '{ "query": "content:orçamento"}'

Solr Schema:

- A Solr Schema defines the configuration of fields and field types for a given core. Default field types include boolean, string, text_general, etc.
- Field types can also be configured based on the default ones.
- Three types of fields can be defined:
    - **Fields** are the specific fields defined — e.g., title of type string and content of type text.
    - **dynamicFields**, are used to index fields not explicitly defined in your schema, i.e. identical to a regular field but application is based on a wildcard — e.g., define all fields ending in "_txt" as text_general.
    - **copyFields**, automatically copy the value of a given field to another. Use case: perform different transformations to ingested values, e.g., remove punctuation from a text but keep the original for displaying.

Solr Schema Type Definition:

- The Schema API is used to set a core's schema definition.
- The schema definition can be provided in JSON format.
    - Reference: Solr: Schema API
- In the next example we define a new type "newsContent" of type TextField.
    - Reference: Solr: Field Types Included with Solr
- Note that to define a new schema, the previous one needs to be deleted.

    - Easiest way is to delete and create.
    - docker exec pri_solr solr delete -c ptnews
- To load a schema defined in a JSON file use:
    - curl -X POST -H 'Content-type:application/json' \ --data-binary "@./ptnews-schema.json" \ http://localhost:8983/solr/ptnews/schema
- And then index the documents.
- Verify the new type definition in Solr Admin.

Analyzers:

- The schema definition can include, for each field type, definitions for:
    - **indexAnalyzer**, transformations to perform as the documents are indexed. These transformations are applied to the indexed terms, not the stored values.
    - **queryAnalyzer**, transformations to perform when queries are processed.
- Analyzers can include one tokenizer and multiple filters.

Tokenizers:

- **Tokenizers break the input text stream into a stream of tokens.**
- Solr built-in tokenizers: Solr: Tokenizers (see example inputs and outputs).
- Example tokenizers:
    - **Standard Tokenizer**, splits the text field into tokens, treating whitespace and punctuation as delimiters.
    - **Lower Case Tokenizer**, tokenizes the input stream by delimiting at non-letters and then converting all letters to lowercase. Whitespace and non-letters are discarded.
    - **N-Gram Tokenizer,** reads the field text and generates n-gram tokens of sizes in the given range.

Filters:

- Filters processes a stream of tokens and generates a different set of tokens.
- Solr built-in tokenizers: Solr: Filters (see in / out examples).
- Example filters:
    - **ASCII Folding Filter,** this filter converts alphabetic, numeric, and symbolic Unicode characters to their ASCII equivalents, if one exists.

    - **Lower Case Filter**, converts any uppercase letters in a token to the equivalent lowercase token. All other characters are left unchanged.
    - **Stop Filter**, this filter discards, or stops analysis of, tokens that are on the given stop words list. A standard stop words list is included in the Solr conf directory, named stopwords.txt, which is appropriate for typical English language text.
    - **Snowball Porter Stemmer Filter**, applied a language-specific stemmer generated by Snowball, a software package that generates pattern-based word stemmers. Includes built-in support for Portuguese.

Solr Query Parsers:

- Different query parsers can be used to match documents to queries.
- The **standard query parser** offers an intuitive syntax but is very strict, i.e. is very intolerant to syntax errors.
- The **DisMax query parser** is designed to through as little errors as possible, being appropriate for consumer facing applications.
- The **Extended DisMax query parser** is an improved version that is both forgiving in the syntax and supports complex query expressions.

eDisMax Query Parser:

- The query parser to use can be defined with the ‘defType’ parameter.
- For both the DisMax and eDisMax query parser the ‘qf’ parameter is available, defining the list of fields where the search should be executed.
    - Instead of using:
        - q=title:flower+AND+content:flower+AND+summary:flower
    - We can simply use:
        - defType=edismax&qf=title+content+summary&q=flower
- A debug parameter is available to debug query execution — debug=all.

Weighting Fields:

- Document fields can be weighted differently, i.e. contribute differently to estimate the relevance of a document.
- The ‘qf’ parameter can be used to specify relative field weights.
    - qf=title^5+content+summary^3

- Additional information to understand ranking decisions can be obtained with ‘debug’ and ‘debug.explain.structured’ parameters.
    - debug=all&debug.explain.structure=true