Chapter IR:I

I. Introduction

- Information Retrieval in a Nutshell
- Examples of Retrieval Problems
- Terminology
- Delineation
- Historical Background
- Architecture of a Search Engine
  - Acquisition
  - Text Analysis
  - Indexing
  - User Interface
  - Query Analysis and Synthesis
  - Retrieval
  - Re-ranking
  - Evaluation

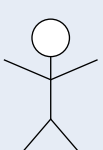
©HAGEN/POTTHAST/STEIN 2023

Remarks:

- Software architecture refers to the high level structures of a software system. These structures are needed to reason about the software system. Each comprises software elements, relations among them, and properties of both elements and relations. [Wikipedia]
- Software architecture can be specified at various levels of abstraction, also called views. We adopt a high-level functional view, showing what a search engine does, not how it is implemented.
- The implementation of a search engine must meet two requirements: effectiveness and efficiency. Effectiveness refers to retrieval quality; efficiency to retrieval speed. Other requirements boil down to these two categories. Examples: Scalability demands efficiency; result freshness improves effectiveness and demands efficiency.
- Search engines basically implement two processes, indexing and retrieval, on top of a storage layer. Indexing is a background process to prepare to-be-searched data for efficient search, as well as updating it. Retrieval offers a user interface for query submission, and implements query analysis and synthesis, and retrieval. The storage layer implements a data model for storing documents, index, and logs so that distributed and parallel search are possible.
- Compare with Google's early architecture described in "The Anatomy of a Large-scale Hypertextual Web Search Engine" by Brin and Page 1998.

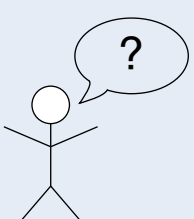
©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

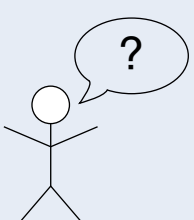


© HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Indexing

Storage

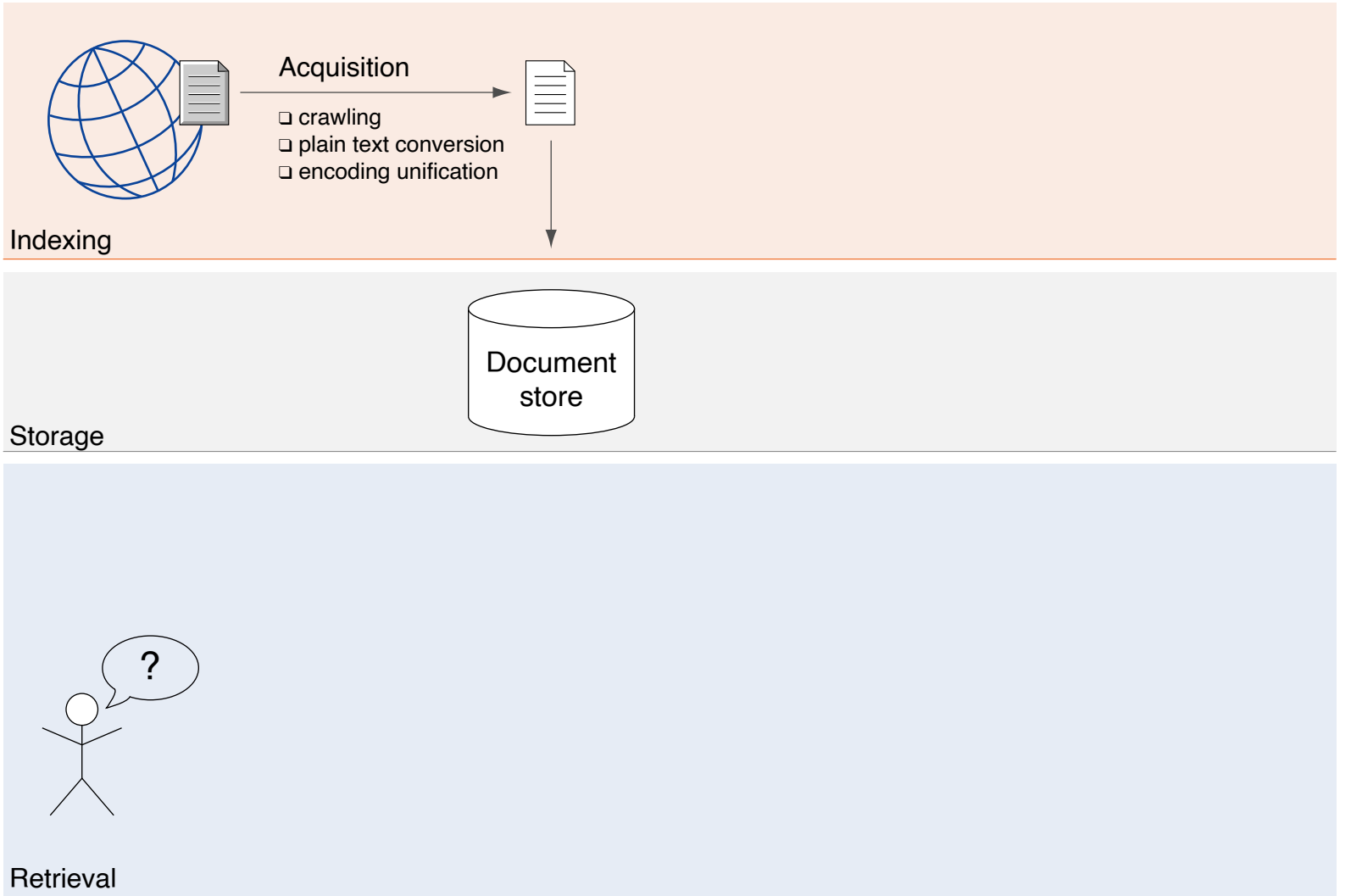


Retrieval

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Acquisition



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Acquisition

In the acquisition step, documents are collected, prepared, and stored.

Key components:

- Crawler
- Converter
- Document Store

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Acquisition: Crawler

A crawler discovers and acquires documents.

Web crawler

- Discovers new web pages via hyperlinks
- What are challenges for web crawling?

Site crawler / focused crawler / topical crawler

- Web crawler for websites / that discards documents (wrt. topics, genres, etc.)
- May exploit structured sitemaps, RSS, or Atom feeds on sites.
- May require a document classifier to identify matching documents.
- Examples: academic search, news search, business search, job search, etc.

Document crawler

- Scans local directories, emails, databases, etc.
- Examples: enterprise search, desktop search

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Acquisition: Crawler

A crawler discovers and acquires documents.

Web crawler

- Discovers new web pages via hyperlinks
- Exploration policy (e.g., avoid spider traps), duplicate identification (e.g., URL normalization), parallelization, revisit / update policy, politeness

Site crawler / focused crawler / topical crawler

- Web crawler for websites / that discards documents (wrt. topics, genres, etc.)
- May exploit structured sitemaps, RSS, or Atom feeds on sites.
- May require a document classifier to identify matching documents.
- Examples: academic search, news search, business search, job search, etc.

Document crawler

- Scans local directories, emails, databases, etc.
- Examples: enterprise search, desktop search

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Acquisition: Converter

A converter unifies documents as follows:

Reformatting / text extraction

- Documents come in a variety of formats. Examples: HTML, PDF, DOC
- Implementation benefits from unified formats (HTML in web search)
- Plain text extraction from documents is lossy (e.g., formatting / layout is lost)
- Layout analysis is still useful for retrieval (see segmentation)
- Document formats are often invalid, they must still be parsed

Encoding normalization

- Plain text documents come in a variety of encodings (e.g., ASCII, Unicode)
- Subsequent processing steps require unified input encoding (e.g., Unicode)
- Encoding specifications are untrustworthy, encodings must be detected
- Documents' encodings are often invalid, they must be repaired

Errors propagate; when visible in search results, the search engine is blamed.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Acquisition: Document Store

The document store manages all documents acquired:

Original and converted documents

- Why is it sensible to mirror documents locally?

Document metadata

- Provenance data, such as origin, crawl date, etc.

Version history

- Every recrawl of a document is kept.
- Older document versions are useful for later analyses.

Scale often demands for a distributed document store.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Acquisition: Document Store

The document store manages all documents acquired:

Original and converted documents

- Original may not always be available
- Fast reprocessing (e.g., when a processing step is improved)
- Fast snippet generation

Document metadata

- Provenance data, such as origin, crawl date, etc.

Version history

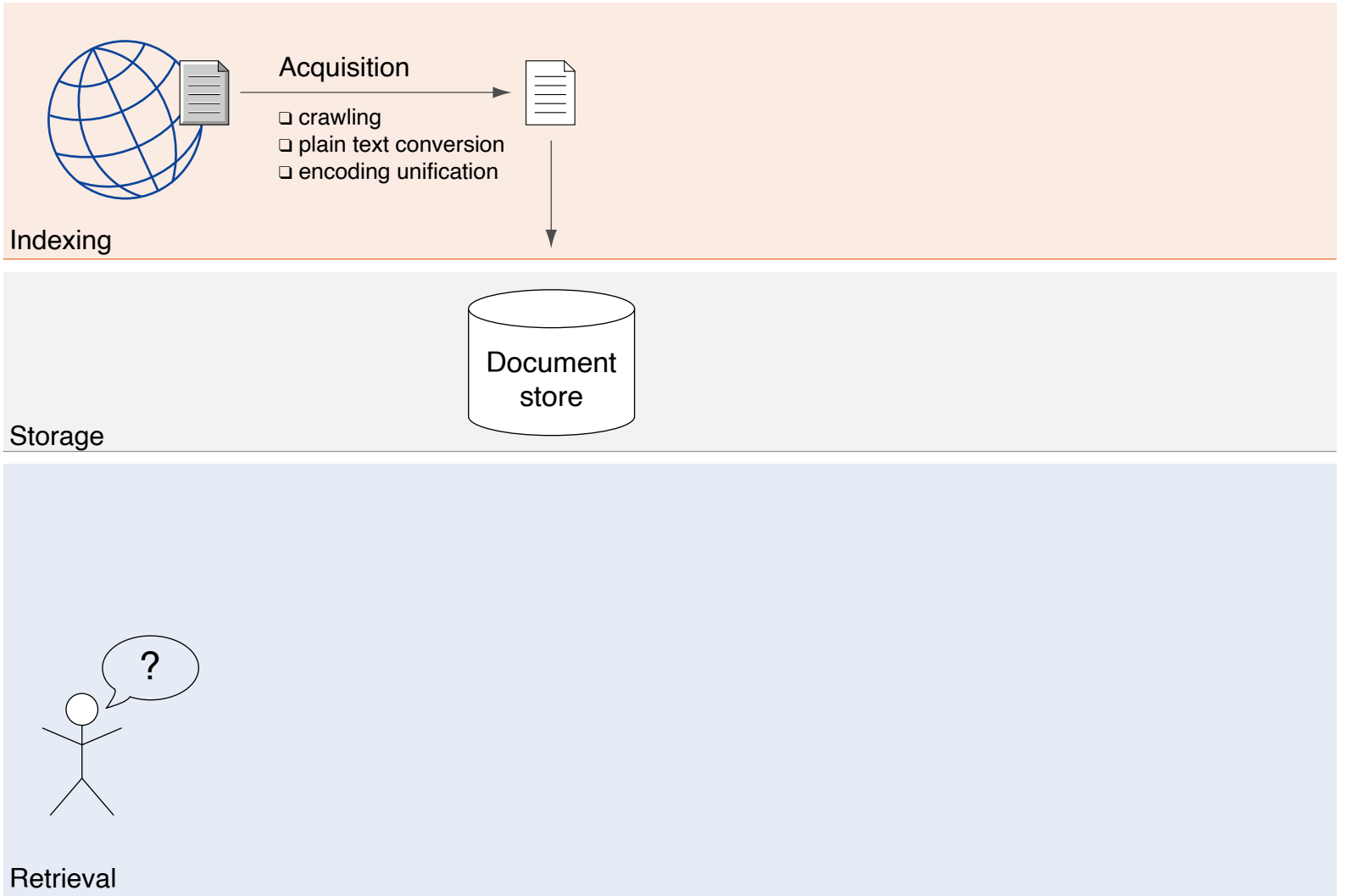
- Every recrawl of a document is kept.
- Older document versions are useful for later analyses.

Scale often demands for a distributed document store.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

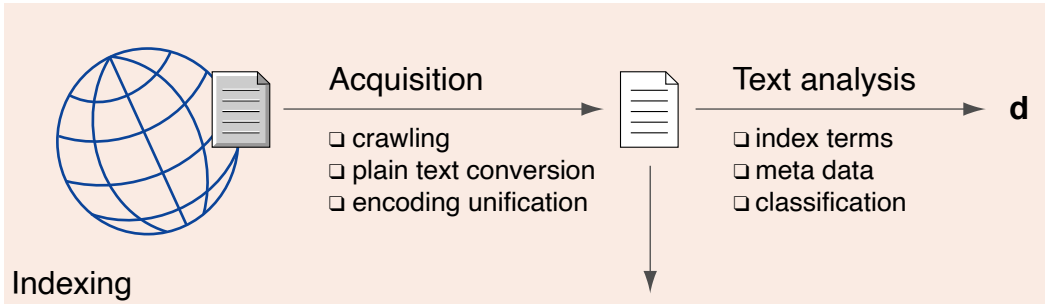
Acquisition



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Text Analysis



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Text Analysis

The text analysis extracts from a document the “keys” by which it can be looked up in the index. Two kinds of keys are distinguished:

Index terms: (terms, for short)

- Words or phrases from a document’s text
- Their purpose is to represent what a document is about
- All index terms of all documents combined form the terminology

Features:

- A feature is a measurable property of a document, a feature set represents it
- Different feature sets are suitable for different classification targets
- Example: language, “spaminess”, topic, genre, . . .

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Text Analysis

The text analysis extracts from a document the “keys” by which it can be looked up in the index. Two kinds of keys are distinguished:

Index terms: (terms, for short)

- Words or phrases from a document’s text
- Their purpose is to represent what a document is about
- All index terms of all documents combined form the terminology

Features:

- A feature is a measurable property of a document, a feature set represents it
- Different feature sets are suitable for different classification targets
- Example: language, “spaminess”, topic, genre, . . .

Key components:

- Segmenter
- Stopping
- Stemmer / Lemmatizer
- Link Extraction
- Information Extraction
- Classification

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Text Analysis: Segmenter

A segmenter breaks down a document into its constituent parts at two levels:

Page segmentation

- Analysis of the HTML source of a web page with regard to its structure
- Extraction of main content vs. ads, navigation, header, footer, etc.
- HTML pages often do not meet the specification (e.g., semantic elements)
- Extraction of text structure and text formatting (e.g., headings, paragraphs)

Text segmentation

- Analysis of a plain text with regard to linguistic units (e.g., words, sentences)
- Tokenization turns a text into a token sequence (token $\approx$ word).
  (a) White space tokenization: tokens are separated by white space characters
  (b) Regular expression tokenization: a token is an alphanumeric string
  – Why are these definitions insufficient?

- Lower-casing of words ensures more matches with query terms

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Text Analysis: Segmenter

A segmenter breaks down a document into its constituent parts at two levels:

Page segmentation

- Analysis of the HTML source of a web page with regard to its structure
- Extraction of main content vs. ads, navigation, header, footer, etc.
- HTML pages often do not meet the specification (e.g., semantic elements)
- Extraction of text structure and text formatting (e.g., headings, paragraphs)

Text segmentation

- Analysis of a plain text with regard to linguistic units (e.g., words, sentences)
- Tokenization turns a text into a token sequence (token $\approx$ word).
  (a) White space tokenization: tokens are separated by white space characters
  (b) Regular expression tokenization: a token is an alphanumeric string
  - Words are not only separated by white space, but also punctuation
  - Contractions and words with special characters are neglected
- Lower-casing of words ensures more matches with query terms

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Text Analysis: Stopping

Stopping (also stop word removal) discards a selection of words from the set of index terms of a document. Candidates for stop words:

- Function words
  Words that carry little semantics, are ambiguous, serve only grammatical purposes, or specify attitude or mood. Examples: the, of, to, for.
- Frequent words
  The most frequently appearing words of a language, or within a collection of documents. Example: “Wikipedia” appears on every Wikipedia page.
- Domain-specific words
  Words that do not discriminate in a given search domain. Example: “learning” may be ignored in the education domain, regardless of its frequency.

- Upsides: reduced index size, faster query processing speed, reduced noise.
- Downsides: many special cases missed. Example: “to be or not to be”.
- Retrieval effectiveness improvement depends on the retrieval model.
- Stopping on index terms is often more conservative than on queries.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Text Analysis: Stemming / Lemmatization

Stemming reduces inflected index terms to a common stem.

Example: “statistics”, “statistic”, and “statistical” refer to basically the same abstract concept.

Two approaches to stemming can be used:

- (Heuristic) Stemming
  Rule-based affix elimination (i.e., suffixes, prefixes, and infixes). Examples: “work*er*”, “*mega*volt”, “un-*bloody*-likely”. Naive heuristic: truncate word after four letters.
- Lemmatization
  Mapping of a word to its root form, even if it is spelled differently. Example: “was” $\rightarrow$ “be”.
- Upside: finding more relevant documents (better retrieval effectiveness)
- What are problems related to aggressive stemming?

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Text Analysis: Stemming / Lemmatization

Stemming reduces inflected index terms to a common stem.
Example: “statistics”, “statistic”, and “statistical” refer to basically the same abstract concept.

Two approaches to stemming can be used:

- (Heuristic) Stemming
  Rule-based affix elimination (i.e., suffixes, prefixes, and infixes). Examples: “work*er*”, “*mega*volt”, “un-*bloody*-likely”. Naive heuristic: truncate word after four letters.
- Lemmatization
  Mapping of a word to its root form, even if it is spelled differently. Example: “was” $\rightarrow$ “be”.

- Upside: finding more relevant documents (better retrieval effectiveness)
- Downsides: conflations of unrelated words, lemmatization is expensive
  Example: “university”, “universe”, and “universal” are mapped to “univers” by common stemmers. Stemmed words may not be real ones. Lemmatization requires dictionaries.
- Retrieval effectiveness improvement varies with language.
  Example: English vs. Arabic. Alternative approach: query expansion.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Text Analysis: Link Extraction

Extraction of links and anchor texts from a document. This serves two purposes:

Link analysis

- Hyperlinks induce a graph among web pages.
- Link analysis traverses this graph to identify authoritative web pages.
- Algorithms for link analysis include PageRank and HITS.

Text augmentation with anchor texts

- The text found on a web page may be insufficient to describe its contents.
- Examples: product pages or pages showing only images.
- Anchor texts as well as the text before and after may justify a link.
- Anchor texts are added to the text extracted from a linked page.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Text Analysis: Information Extraction

Information extraction identifies more complex index terms by means of natural language processing technology:

- **Noun phrases**
  Phrases which have a noun as their head word (i.e., a noun and any word that modifies it).
  Examples: "*The yellow house* is for sale.", "I want *a skateboard*."
- **Named entities**
  Words or phrases that designate something in the "real" world.
  Examples: places, people, organizations, etc.
- **Coreference resolution**
  Coreferences (i.e., anaphora and cataphora) are expressions that refer backward or forward in a text, respectively. Resolving *them* is important for text understanding, yet, still unsolved.
- **Relation detection**
  Extraction of relations between named entities mentioned in the text.
  Example: "*Bill* lives in the *USA*. *He* works at the *United Nations*."
- **Semi-structured information extraction**
  Extraction and analysis of tables, quotes, references, comments, etc.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Text Analysis: Classification

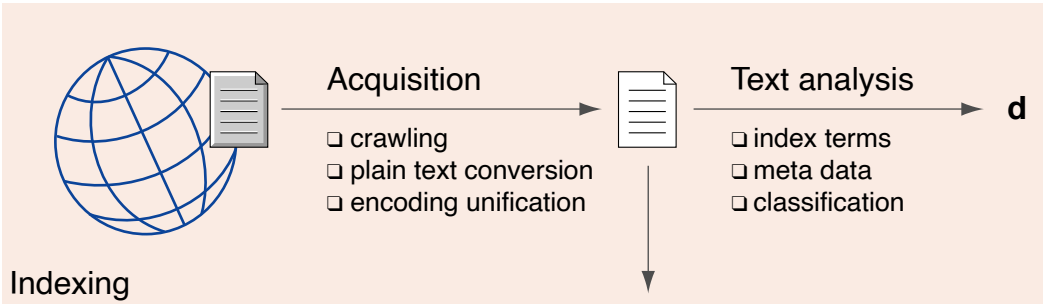
Machine learning is applied to classify or categorize documents based on features. Common classification goals are:

- Language identification
  Determines the (main) language of a web page.
- Spam detection / malware detection
  Determines if a website/web page is trying to undermine a search engine's ranking (spam), or harm its users (malware).
- Topic categorization
  Determines the topic of a document. Topics overlap, form hierarchies, and depend on the search domain. Examples: sports, politics, technology, etc., or user-/domain-specific ones.
- Cluster analysis
  Determines previously unknown topics; automatic cluster labeling required.
- Genre categorization
  Determines the genre of a web page. Genres overlap, form hierarchies, and depend on the search domain. Examples: personal home page, message board, blog, shop, etc.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

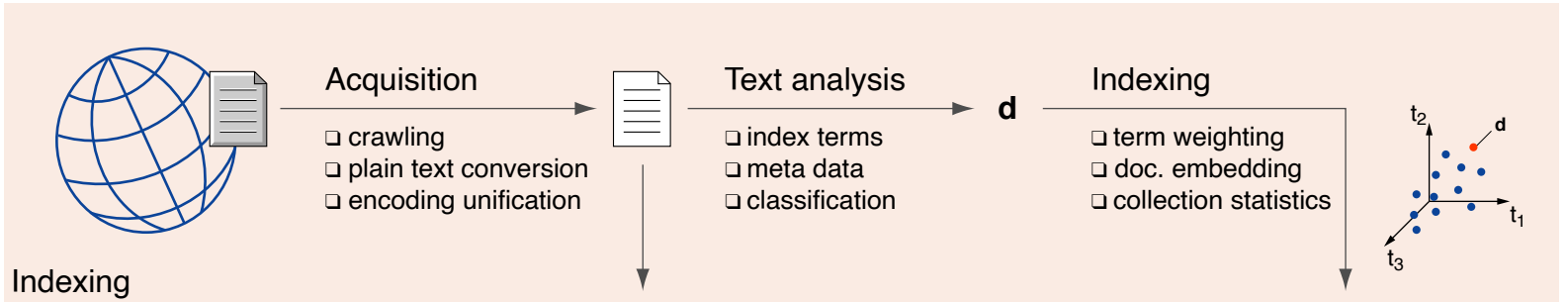
Text Analysis

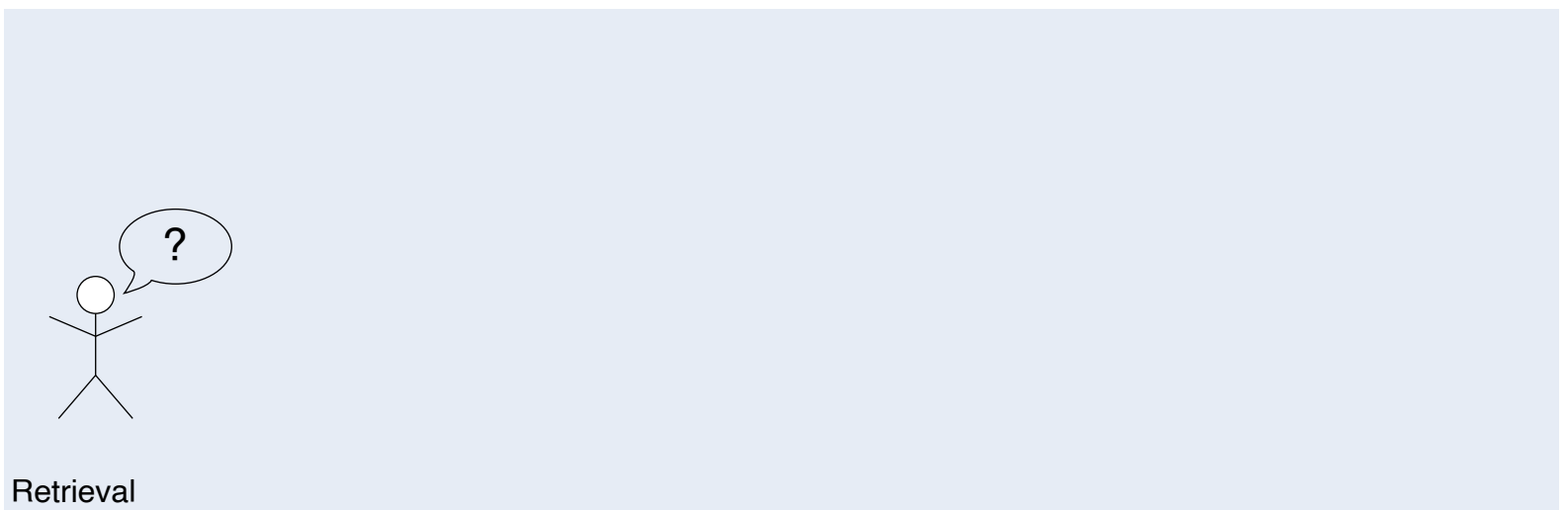


©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Indexing





©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Indexing

The indexing step creates the index data structures required for fast retrieval from the set of acquired documents.

The most commonly used data structure is the inverted index.
Suffix array and signature file are examples of further data structures for special purposes.

Key components:

- Term Weighting
- Index Construction
- Distribution
- Document Statistics

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Indexing: Term Weighting

For each index term of a document, calculate a weight indicating its importance with respect to the document. Basic term weighting schemes:

- Term frequency (*tf*)
  Logarithm of the (normalized) number of occurrences of a term in a document.
- Inverse document frequency (*idf*)
  - Document frequency (*df*): Number of documents containing a term
  - Logarithm of the total number of documents divided by *df*.
- $tf \cdot idf$
  One of the most well-known term weighting schemes in IR.
- BM25
  Similar to $tf \cdot idf$, but yields better retrieval effectiveness.

Term weights are pre-computed and stored in the index to speed up relevance scoring during retrieval.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Indexing: Index Construction

Index construction creates an inverted index data structure by inverting the document–term data to term–document data.

T	$\rightarrow$	**Postings** (Posting Lists, Postlists)			
t_1	$\rightarrow$	$d_1, w_{1,1}$	$d_2, w_{1,2}$		
t_2	$\rightarrow$	$d_1, w_{2,1}$	$d_2, w_{2,2}$	$d_4, w_{2,4}$	
t_3	$\rightarrow$	$d_1, w_{3,1}$	$d_2, w_{3,2}$	$d_4, w_{3,4}$	$d_5, w_{3,5}$
t_4	$\rightarrow$	$d_2, w_{4,2}$			
t_5	$\rightarrow$	$d_1, w_{5,1}$			
$\vdots$					

- Bulk indexing
  Creates an index offline by processing all acquired documents. Once ready, the currently used index is replaced with the new one.
- Index update
  Updates a currently used index online with new documents as they appear.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Indexing: Distribution

Large indexes are distributed (sharded, partitioned) across machines:

Document distribution

- Split collection; smaller indexes for sub-collections on different machines

Term distribution

- Split the index for the entire collection by terms
- Different machines serve different terms

Replication

- Copies of (parts of) indexes at multiple sites

What are arguments for sharding by documents / terms, and for replication?

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Indexing: Distribution

Large indexes are distributed (sharded, partitioned) across machines:

Document distribution

- Split collection; smaller indexes for sub-collections on different machines
- Enables parallelism for indexing and query processing
- Small indexes are faster due to caching; more search results can be retrieved

Term distribution

- Split the index for the entire collection by terms
- Different machines serve different terms
- Not all machines have to process every query

Replication

- Copies of (parts of) indexes at multiple sites
- Reduced delays during query processing
- Fault tolerance

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Indexing: Document Statistics

Auxiliary information about documents is gathered for fast online query processing:

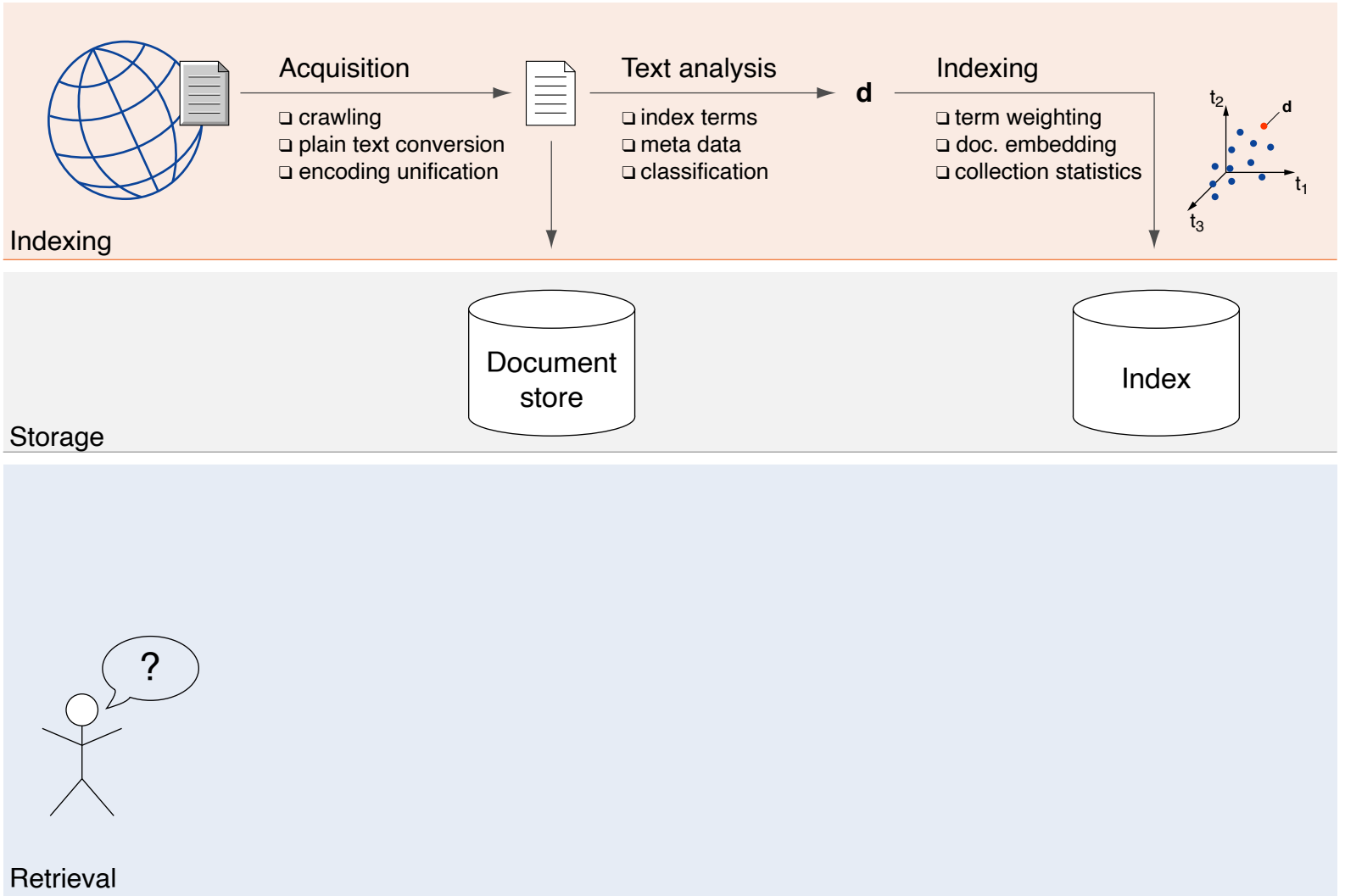
- Term frequencies per document, topic, and genre
  For instance, to compute query-dependent relevance scores.
- Term positions per document
  For instance, to speed up snippet generation.
- Document frequencies per term
  The number of documents a term occurs in; required for relevance scoring.
- Document lengths
  Required for relevance scoring.

The data structure used is a basic key–value store (i.e., a hash map).

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Indexing



©HAGEN/POTTHAST/STEIN 2023

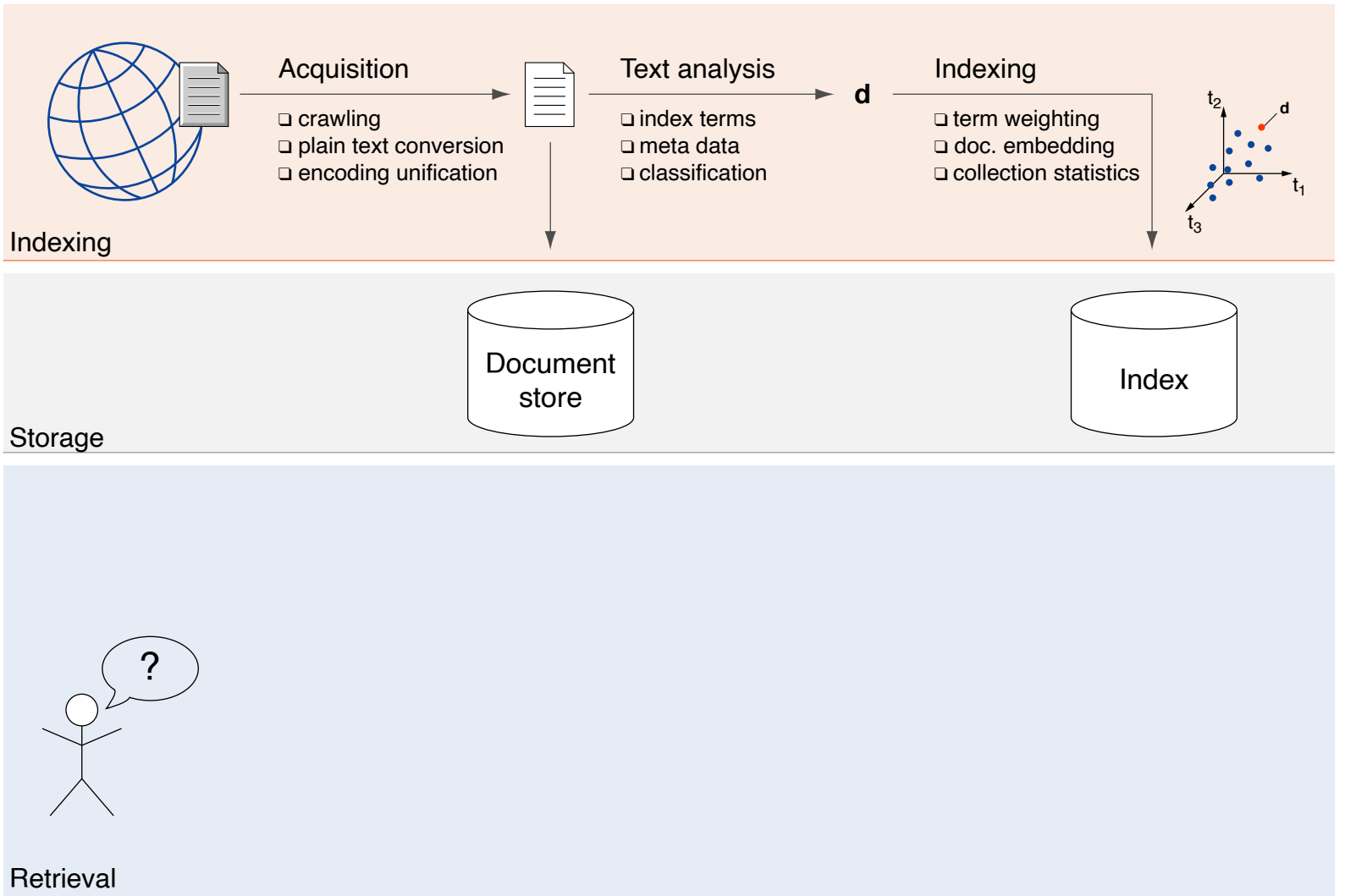
Chapter IR:I

I. Introduction

- Information Retrieval in a Nutshell
- Examples of Retrieval Problems
- Terminology
- Delineation
- Historical Background
- Architecture of a Search Engine
  - Acquisition
  - Text Analysis
  - Indexing
  - User Interface
  - Query Analysis and Synthesis
  - Retrieval
  - Re-ranking
  - Evaluation

©HAGEN/POTTHAST/STEIN 2023

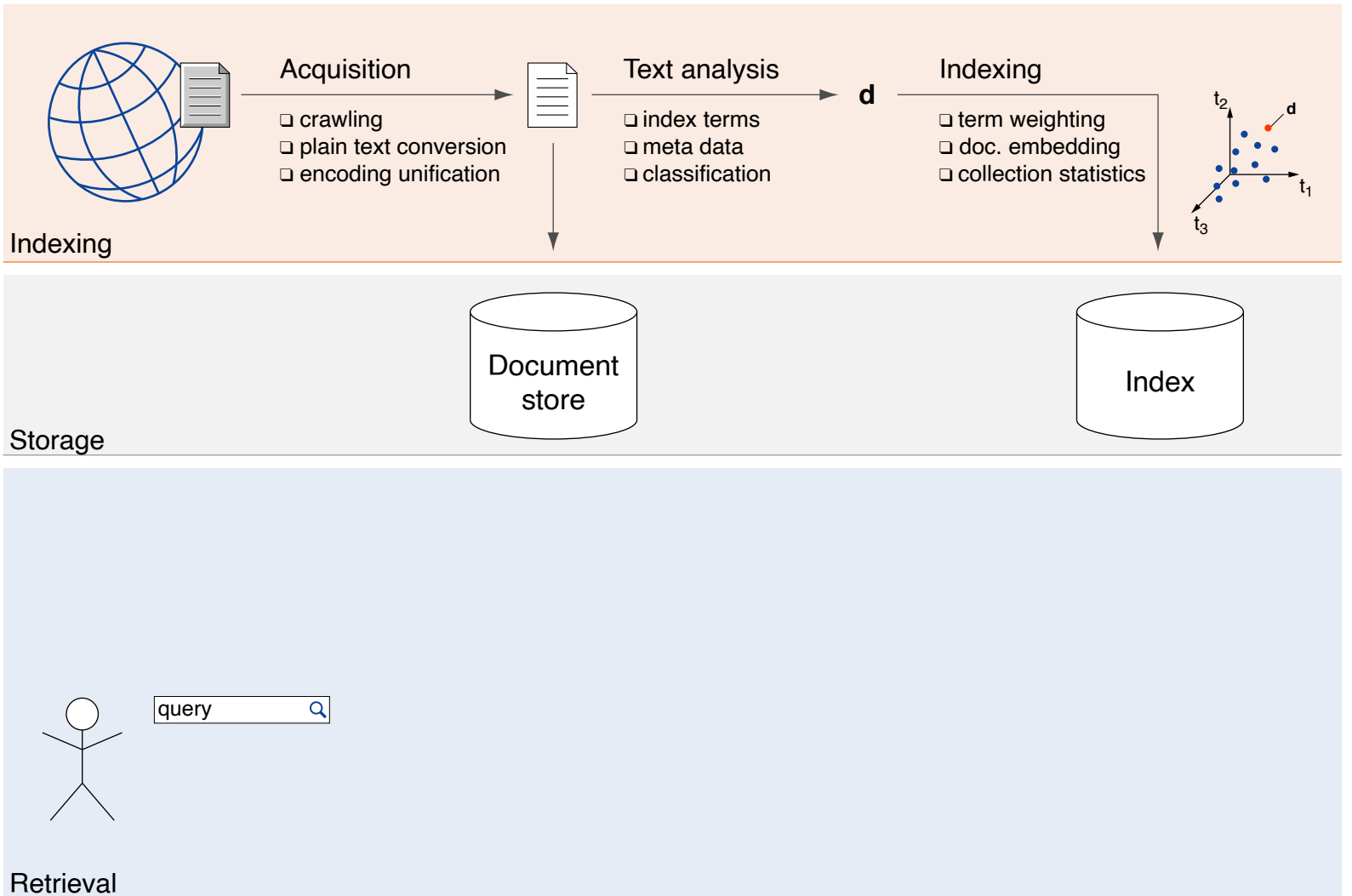
Architecture of a Search Engine



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

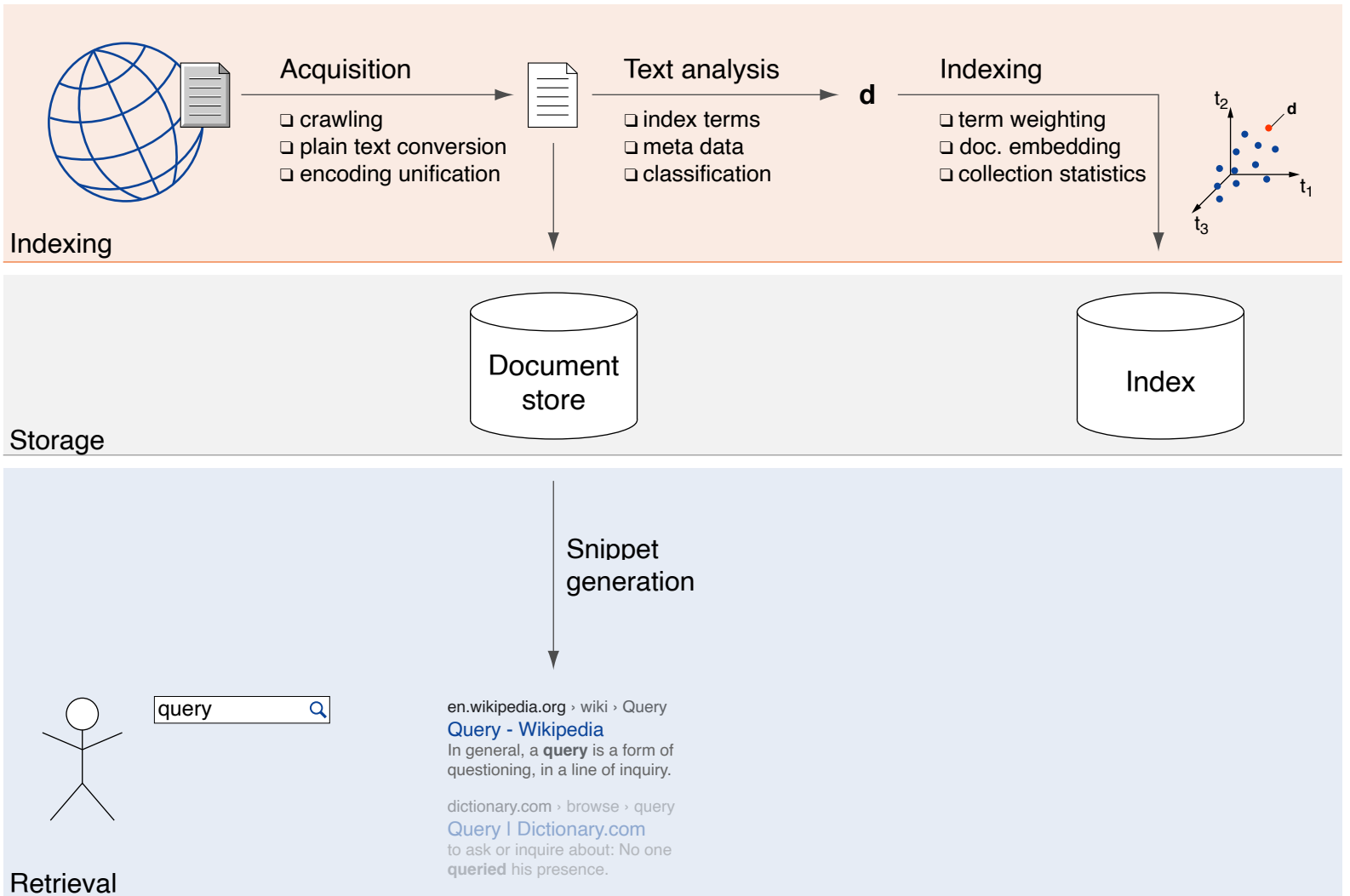
User Interface



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

User Interface



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

User Interface

The user interface of a search engine allows the user to interact with the system. Different interfaces are designed for different usage scenarios.

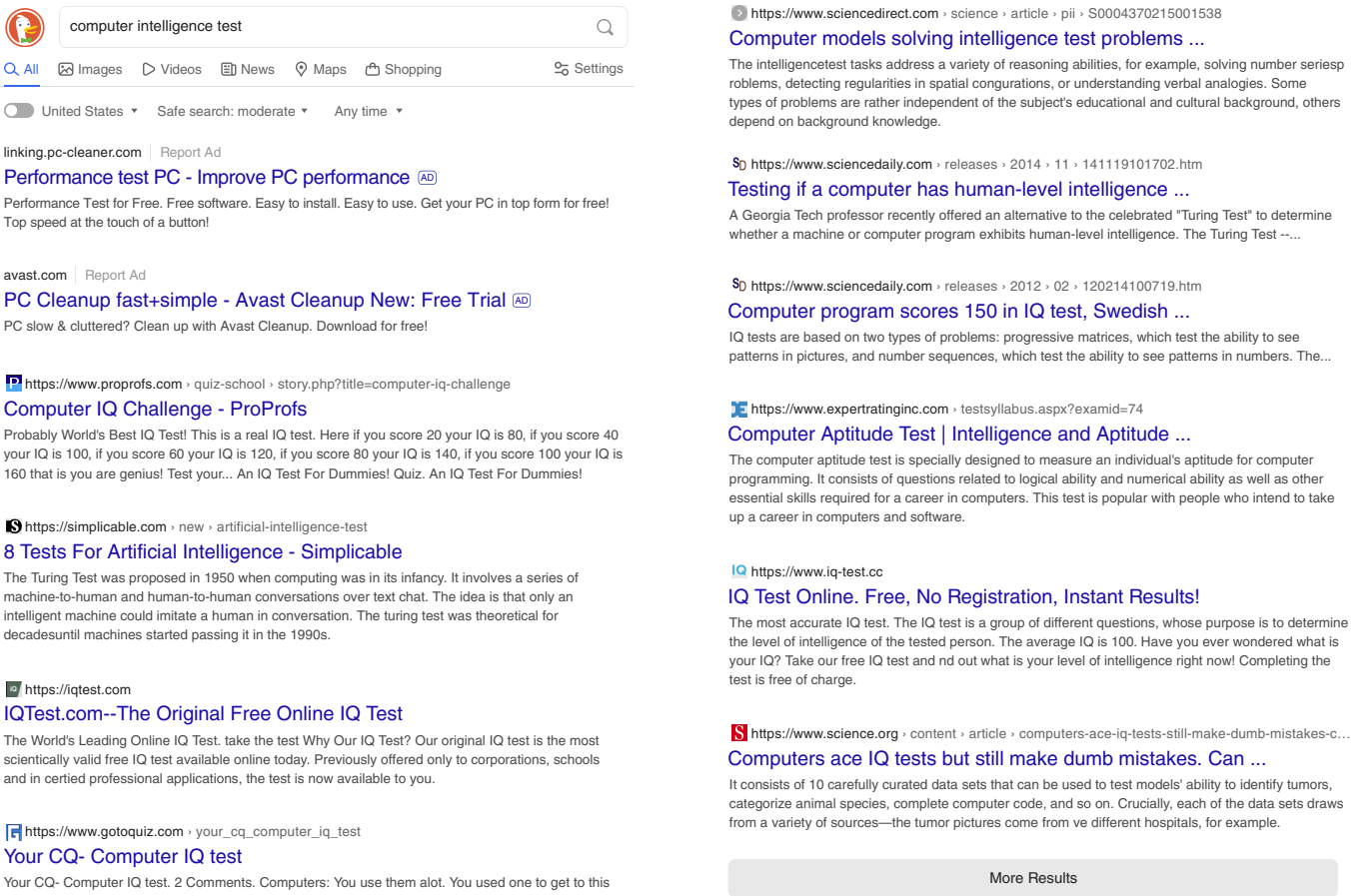
Key components:

- Query Language
- Result Presentation
- Conversational Search

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

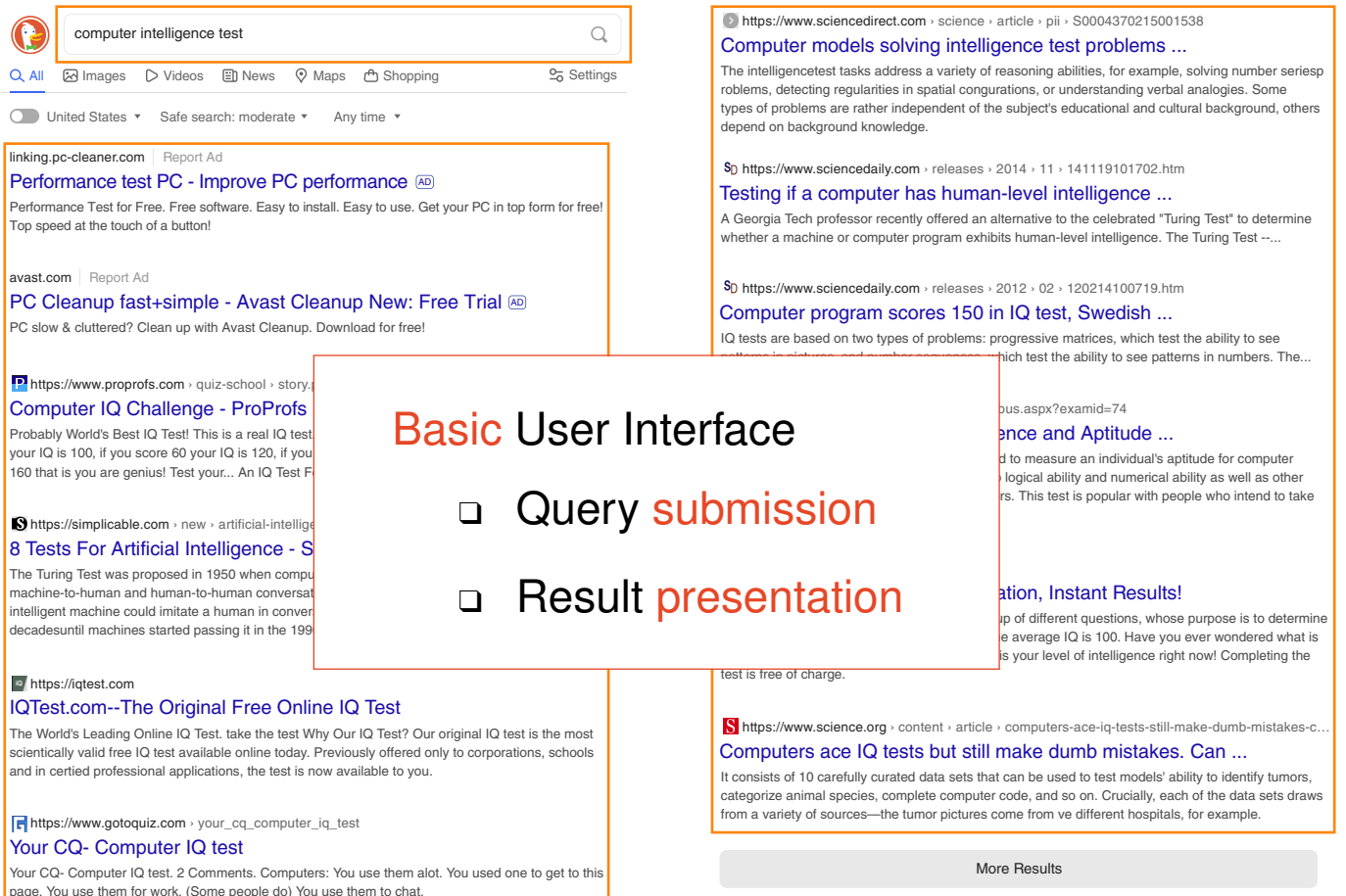
User Interface: Example



©HAGEN/POTTHAST/STEIN 2023

User Interface

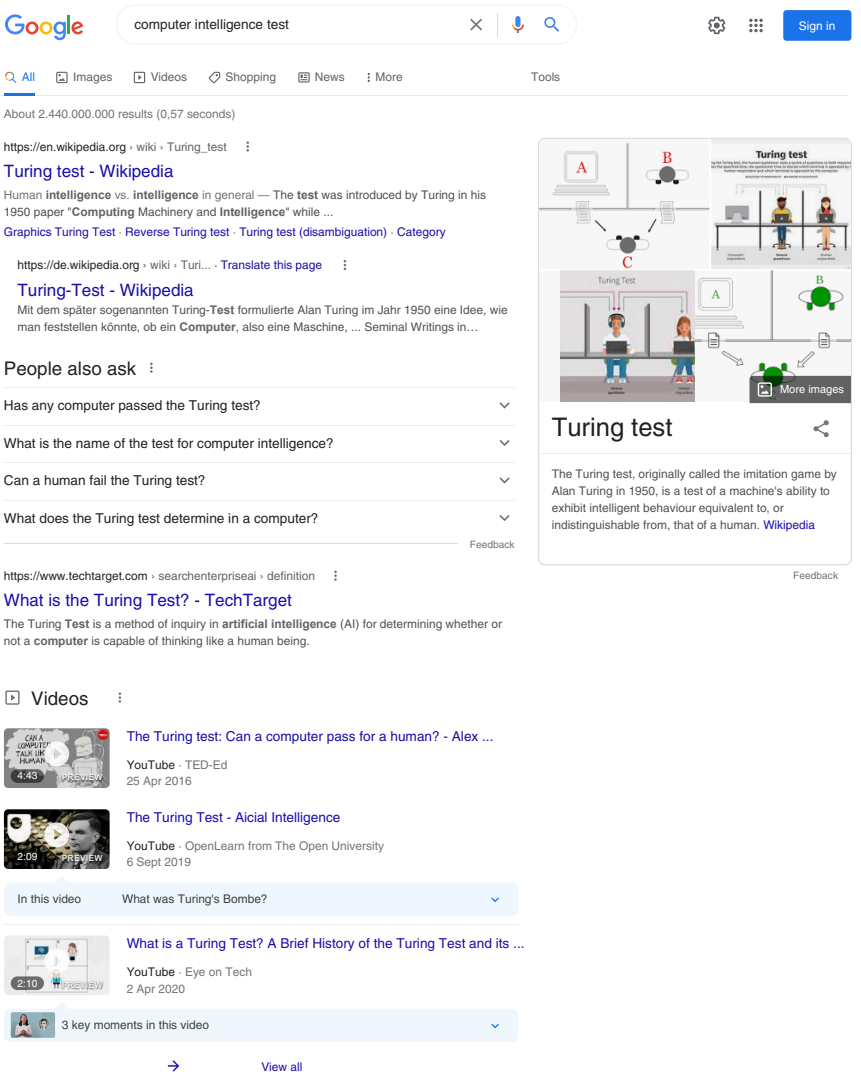
User Interface: Example



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

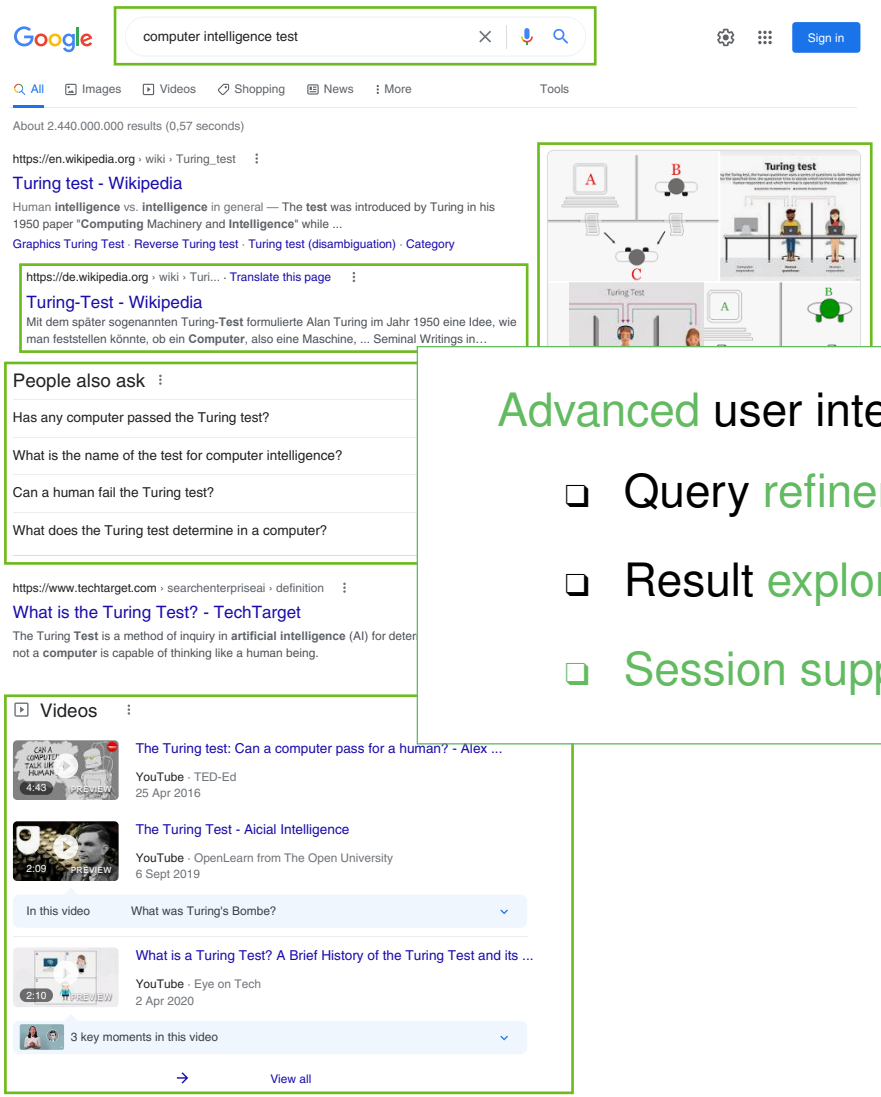
User Interface: Example



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

User Interface: Example



Advanced user interface:

- Query refinement
- Result exploration
- Session support

©HAGEN/POTTHAST/STEIN 2023

Remarks:

- Despite the obvious differences between the two exemplary search engines, neither is DuckDuckGo's user interface extremely simple nor is Google's extremely advanced. On the one hand, DuckDuckGo implements features like search suggestions and the display of related searches. On the other hand, Google's search interface offers little support for exploration nor for search sessions or search missions.
- Many different user interfaces for search engines have been proposed and developed. [Kules et al., 2008]

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

User Interface: Query Language

The query language defines the syntax and semantics of valid queries. It may include commands to influence the search, so-called query operators.

Common query types:

- Structured query
- Keyword query
- Question query
- Query by example

Common operators:

- Boolean operators (AND, OR, NOT (or –))
- Which Boolean operator is implicitly assumed in a keyword query (e.g., in web search)?
- What other operators do you know?

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

User Interface: Query Language

The query language defines the syntax and semantics of valid queries. It may include commands to influence the search, so-called query operators.

Common query types:

- Structured query
- Keyword query
- Question query
- Query by example

Common operators:

- Boolean operators (AND, OR, NOT (or –))
- Quotes / phrasal search (“phrase of text”)
- Field search (title, text, url)
- Wildcards (*, ~synonym)
- Site search (site:example.com)

The most basic form of a query language is the keyword search.

Only about 1% of web queries contain operators. [White and Morris, 2007]
Web search engines cannot expect users to be experts of the query language.

Domain-specific search engines often have specialized query languages, allowing for fine-grained control of retrieval behavior.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

User Interface: Result Presentation

Results are shown in ranking order on the search engine results page (SERP). Several additional processing steps are required to compile the page:

- **Snippet generation**
  Accesses the stored original web page and extracts sentences and phrases that summarize it, dependent on the query. The query's terms are highlighted in the snippet, including alternative inflections or synonyms.

  en.wikipedia.org › wiki › Query
  Query - Wikipedia
  In general, a **query** is a form of questioning, in a line of inquiry.

- **Universal search** (e.g., oneboxes)
  Determines whether other specialized search engines can supply relevant results. Ranks the oneboxes into the SERP as per their importance compared to the organic web results.

- **Ad retrieval**
  Accesses an internal ad search engine tailored to the retrieval of ads relevant to a query from all ads offered by advertisement partners.

- **Facets / Categorization**
  If metadata about documents is available, a side bar that allows to set constraints about the metadata can be displayed. Example: Shopping search engines. Alternatively, documents may be categorized using cluster analysis and cluster labeling.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

User Interface: Conversational Search

ChatNoir

Bonjour! I'm ChatNoir, your friendly cat assistant. How can I help you today?

How can a computer's intelligence be tested?

ChatNoir

To test computer intelligence, you can give it problems to solve and evaluate how it performs. You can also give it tasks that require it to think outside the box, such as creating its own algorithms or coming up with innovative solutions to problems. Additionally, you can give it tests that measure its knowledge and understanding of a given subject.

This is outlined in [1], which also covers other definitions and tests of machine intelligence proposed, such as [2] which provides an overview of these definitions and tests. Finally, [3] is a standard test for machine intelligence.

www.wired.com/ ... /pass-turing-ai-test/

[1] **How to** Pass the Turing Artificial **Intelligence Test** | WIRED

ClueWeb22 · Crawled Aug 2022

How to Pass the Turing Artificial **Intelligence Test** | WIRED WIRED Staff Science Jun 19, 2012 2:00 PM How to Pass the Turing Artificial **Intelligence Test** Are you human or a **machine**? Prove it, by passing the Turing Test — a test of the ability of a **machine to** exhibit intelligent behavior.

www.shivonzilis.com/

[2] **Machine Intelligence** - Shivon Zilis

ClueWeb22 · Crawled Aug 2022

Teams need **to** figure out **how to test** newfound capabilities, and applications need **to** change so they don't just a system of record; they also need **to** coach employees and learn from the data they enter.

www.geeksforgeeks.org/turing-test-artificial-intelligence/

[3] Turing **Test** in Artificial **Intelligence** - GeeksforGeeks

ClueWeb22 · Crawled Aug 2022

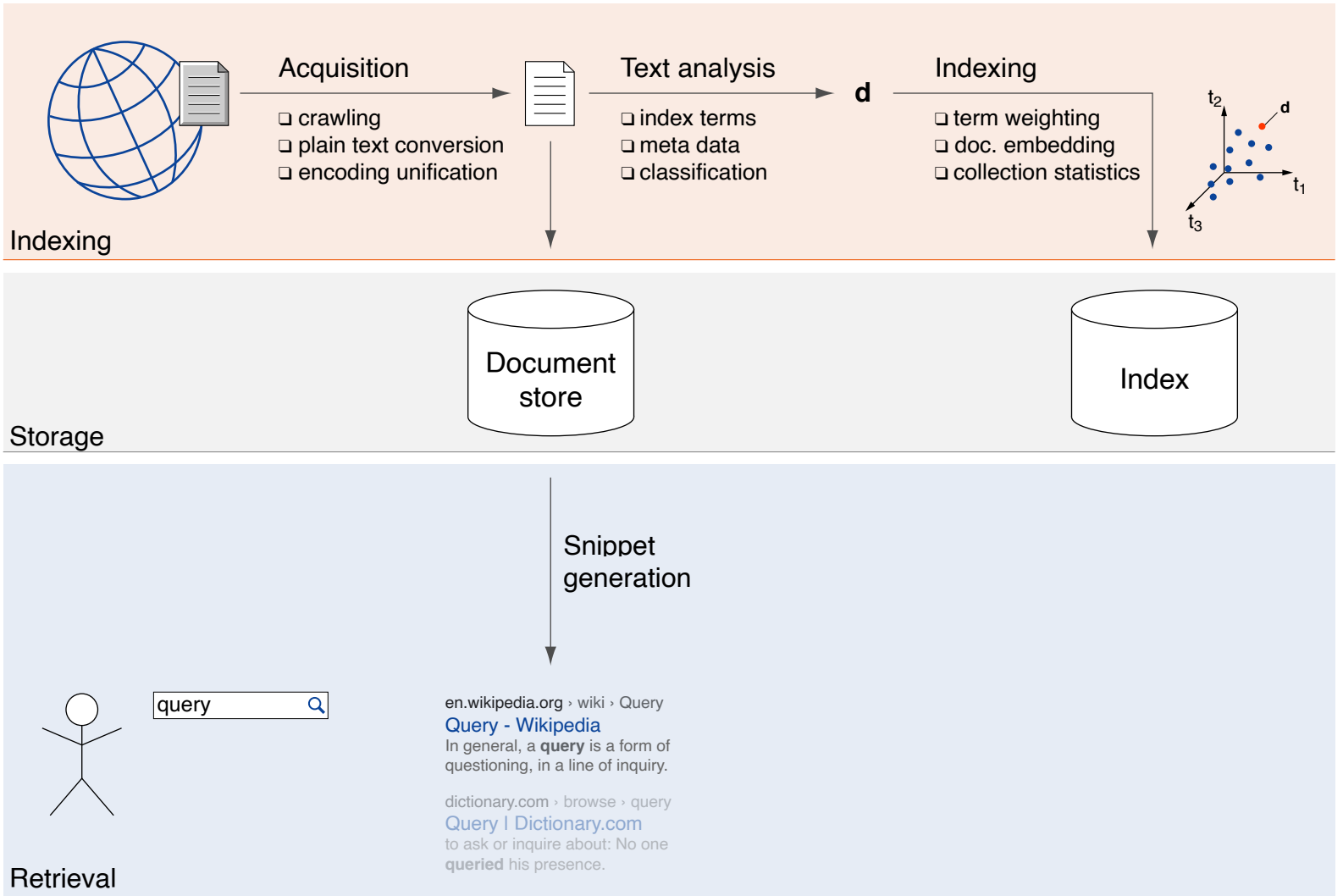
If a **machine** engages in a conversation with a human **how to** process the data it has been demonstrated that a **machine**, He has proposed the following skills of the **test** as follows: The turning judges the conversation of the humans.

Show more...

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

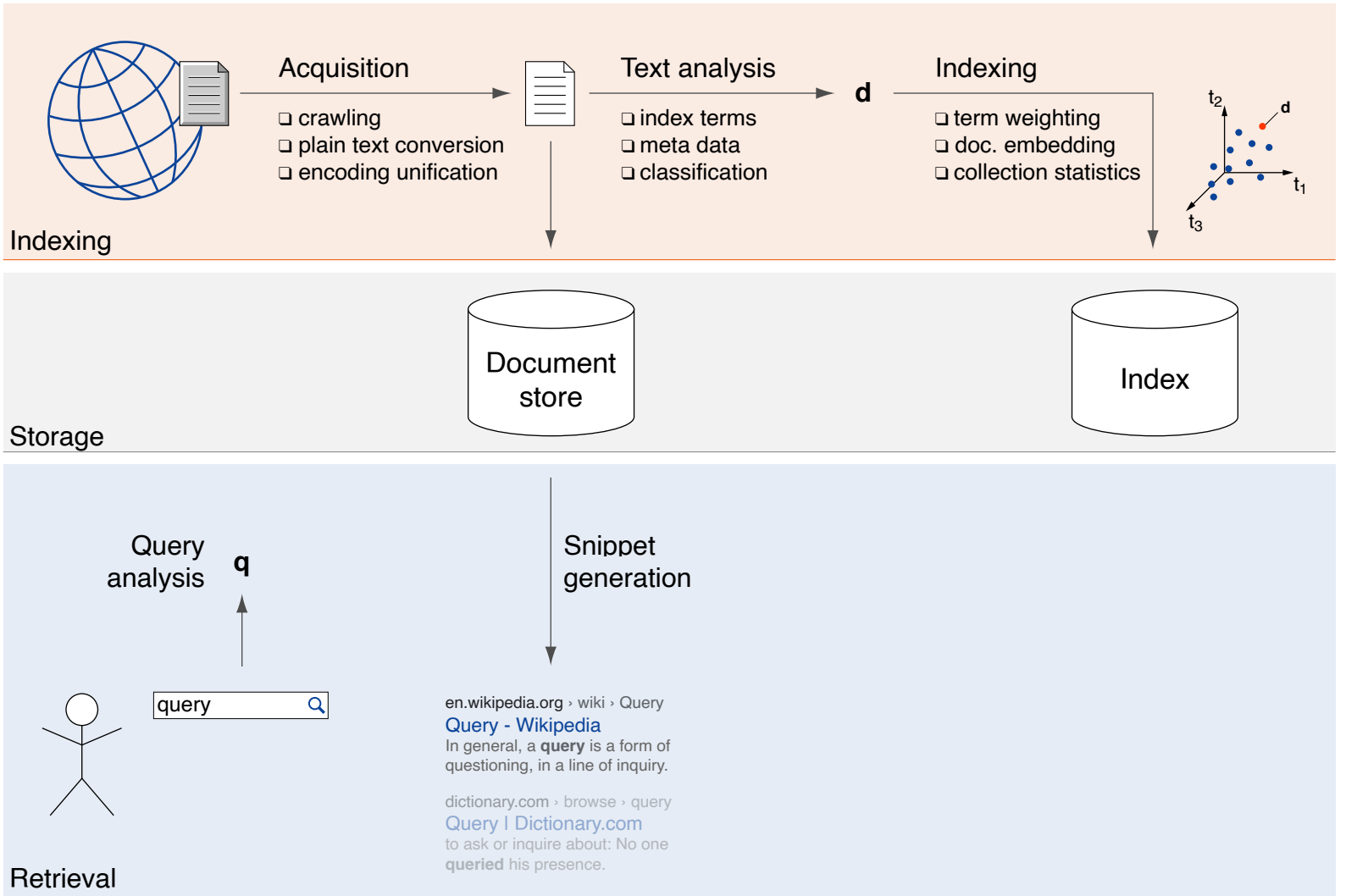
User Interface



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

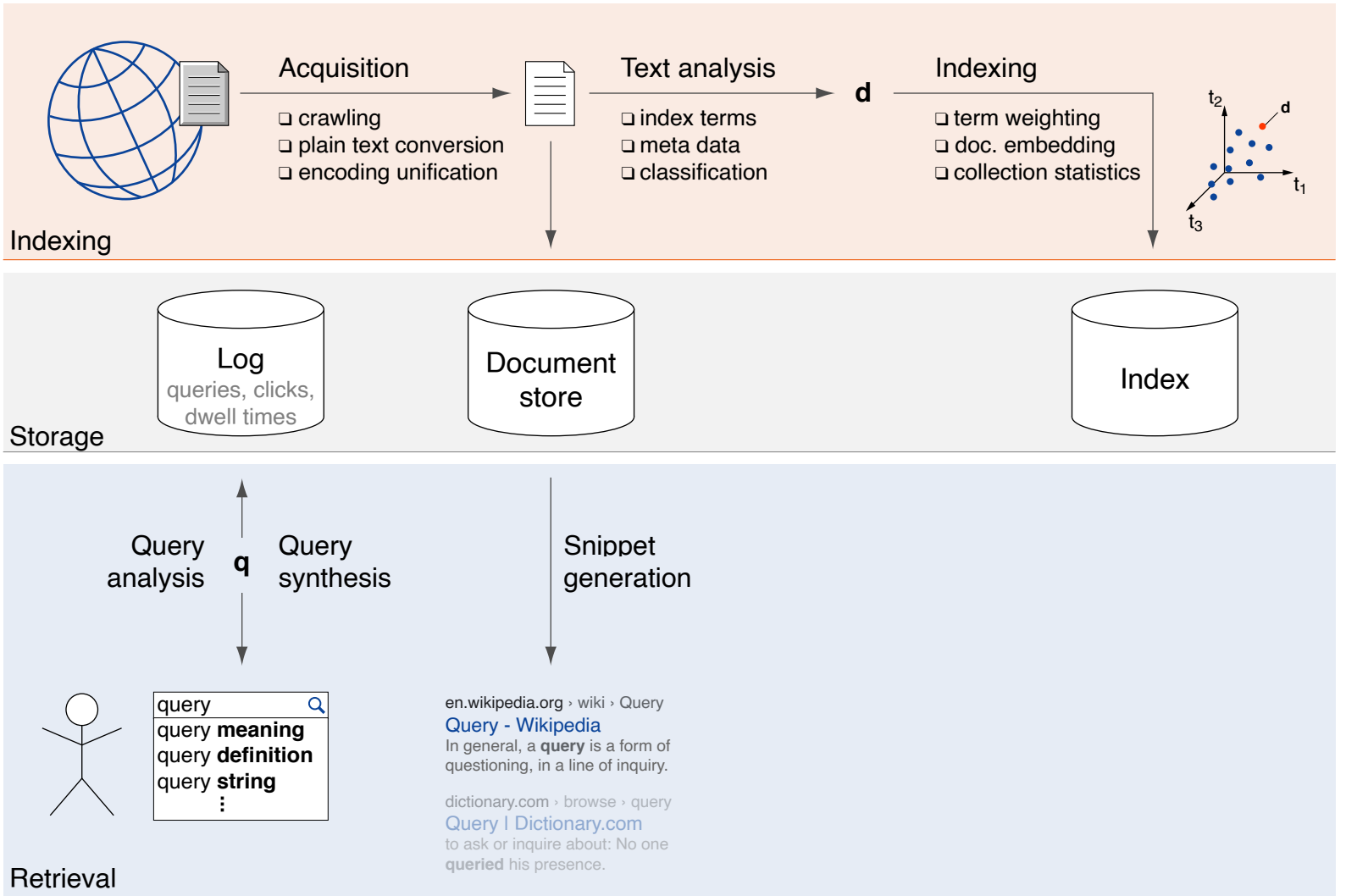
Query Analysis



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Query Analysis and Synthesis



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Query Analysis and Synthesis

Query analysis (query understanding) maps the keywords of a query to the index terms of the terminology to enable retrieval.

Query synthesis generates a new query to represent the user's information need.

Key components:

- Text Analysis
- Logging
- Query Rewriting
- Query Expansion

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Query Analysis and Synthesis: Text Analysis

Query analysis employs a text analysis pipeline similar to that used for documents:

- **Tokenization**
  Tokenization turns a query string into a sequence of tokens.
- **Stopping**
  Stopping, also stop word removal, discards a selection of tokens from the set of query terms.
- **Stemming and lemmatization**
  Stemming aims at reducing inflected index terms to a common stem. Lemmatization maps a word to its root form independent of its spelling.
- **Named entity recognition**
  Recognition of words or phrases that designate something in the “real” world.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Query Analysis and Synthesis: Logging

A search engine keeps logs of user interactions:

- Queries
  Every query submitted to the search engine.
- Result clicks
  Every click on a search result.
- Page interactions
  Data reflecting user behavior on search results pages.
- User tracking
  Association of all of the above data with individual users.

The logs are employed to improve retrieval by synthesizing a better query based on a user's query, and to analyze user experience.

Logs are the most valuable data a search engine collects.

©HAGEN/POTTHAST/STEIN 2023

Remarks:

- Further applications of the log include user experience analysis and optimization. The user behavior on the search engine's web pages allows for conclusions about its efficacy in supporting the user.

  Example: Google's optimization of its result link color. A designer, Jamie Divine, had picked out a blue that everyone on his team liked. But a product manager tested a different color with users and found they were more likely to click on the toolbar if it was painted a greener shade. As trivial as color choices might seem, clicks are a key part of Google's revenue stream, and anything that enhances clicks means more money. Mr. Divine's team resisted the greener hue, so Marissa Mayer split the difference by choosing a shade halfway between those of the two camps. Her decision was diplomatic, but it also amounted to relying on her gut rather than research. Since then, she said, she has asked her team to test the 41 gradations between the competing blues to see which ones consumers might prefer. [nytimes.com]

- In 2006, the internet company AOL released a large amount of user search requests to the public [Wikipedia]. Though thought to be sufficiently anonymized, this turned out to not be the case; a lot of personal information could be obtained from the log, and a number of users be personally identified just from their sequence of submitted queries. One particular user's story has been turned into an artistic movie called "I love Alaska" in which the sequence of queries over a long time spells out a tragic personal history.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Query Analysis and Synthesis: Query Rewriting

Query rewriting changes the query to improve its chances of retrieving relevant documents. Changes may be suggested to users, or made on-the-fly.

Replacement of terms:

- Query suggestion
  Feedback at various degrees of urgency, ranging from hints to replacement, dependent on confidence. Hints: autocompletion while typing, related queries, “Did you mean . . . ” at Google. Replacement: “Showing results for . . . ” at Google.
- Spelling correction
  A language model predicts the probability of a query, and an error model that of an observed misspelling, given an intended query.

Addition of terms:

- Query expansion
  Inclusion of additional terms to a query. Query logs and term co-occurrences in documents are exploited here. Key approach: relevance feedback.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Query Analysis and Synthesis: Query Rewriting (continued)

Query rewriting changes the query to improve its chances of retrieving relevant documents. Changes may be suggested to users, or made on-the-fly.

Removal of terms:

- Query relaxation
  Removal or optionalization of query terms that appear to render a query overspecific, e.g., when too few documents are retrieved. Example: removal of modifiers in a noun phrase.

Query specialization:

- Query segmentation
  Identification of (alternative) phrases and multi-term concepts in a query to enable diversified retrieval. Example query: `new york times square dance`
- Query scoping
  Focusing of (parts of) a query on specific fields of a document, e.g., its title, or body.
- Personalization
  User profiles allow for tailoring search results to the user's context and interests.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Query Analysis and Synthesis: Query Expansion

The addition of terms to a query so as to retrieve more documents relevant to the user's information need.

Abbreviations:

- Dictionary-based
  Add long forms of abbreviations in a query found in a dictionary.
  Problem: ambiguity (e.g., "st." $\rightarrow$ "saint" or "street"?).
- Machine learning-based
  Mining of abbreviations in context from document collections as ground truth. Recognition in queries, and addition of their long form dependent on query context.

Synonyms, hypernyms, hyponyms:

- Dictionary-based
  Add synonyms of query terms found in a dictionary or thesaurus.
  Problems: diversity, exactness (e.g., "computer" $\rightarrow$ "laptop", "web" $\rightarrow$ "internet").
- Machine learning-based
  Training of word embeddings based on large document collections, computation of word similarity, and usage of all words similar to a query's terms that exceed a similarity threshold.

©HAGEN/POTTHAST/STEIN 2023

Remarks:

- A possible mnemonic to keep in mind that there actually is a difference between the Web (content: web pages, links, etc.) and the Internet (infrastructure: protocols, cables, servers, routers, etc.) are photos from the W3C20 Anniversary Symposium 2014 where the inventors Sir Tim Berners-Lee (Web) and Vinton G. Cerf (Internet) did wear t-shirts that explain who did (not) invent what.

[W3C20 Anniversary Symposium]

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Query Analysis and Synthesis: Relevance Feedback

Relevance feedback refines a query in multiple retrieval runs:

1. Retrieval of documents using a given query.
2. Identification of (non-)relevant documents among the top-ranked ones.
3. Extraction of terms (un-)related to the query's implied information need from the identified documents.
4. Addition of related terms to the query (weight decrease of unrelated terms). Unless a stopping criterion is met, continue at Step 1.

What are possible sources of relevance feedback?

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Query Analysis and Synthesis: Relevance Feedback

Relevance feedback refines a query in multiple retrieval runs:

1. Retrieval of documents using a given query.
2. Identification of (non-)relevant documents among the top-ranked ones.
3. Extraction of terms (un-)related to the query's implied information need from the identified documents.
4. Addition of related terms to the query (weight decrease of unrelated terms). Unless a stopping criterion is met, continue at Step 1.

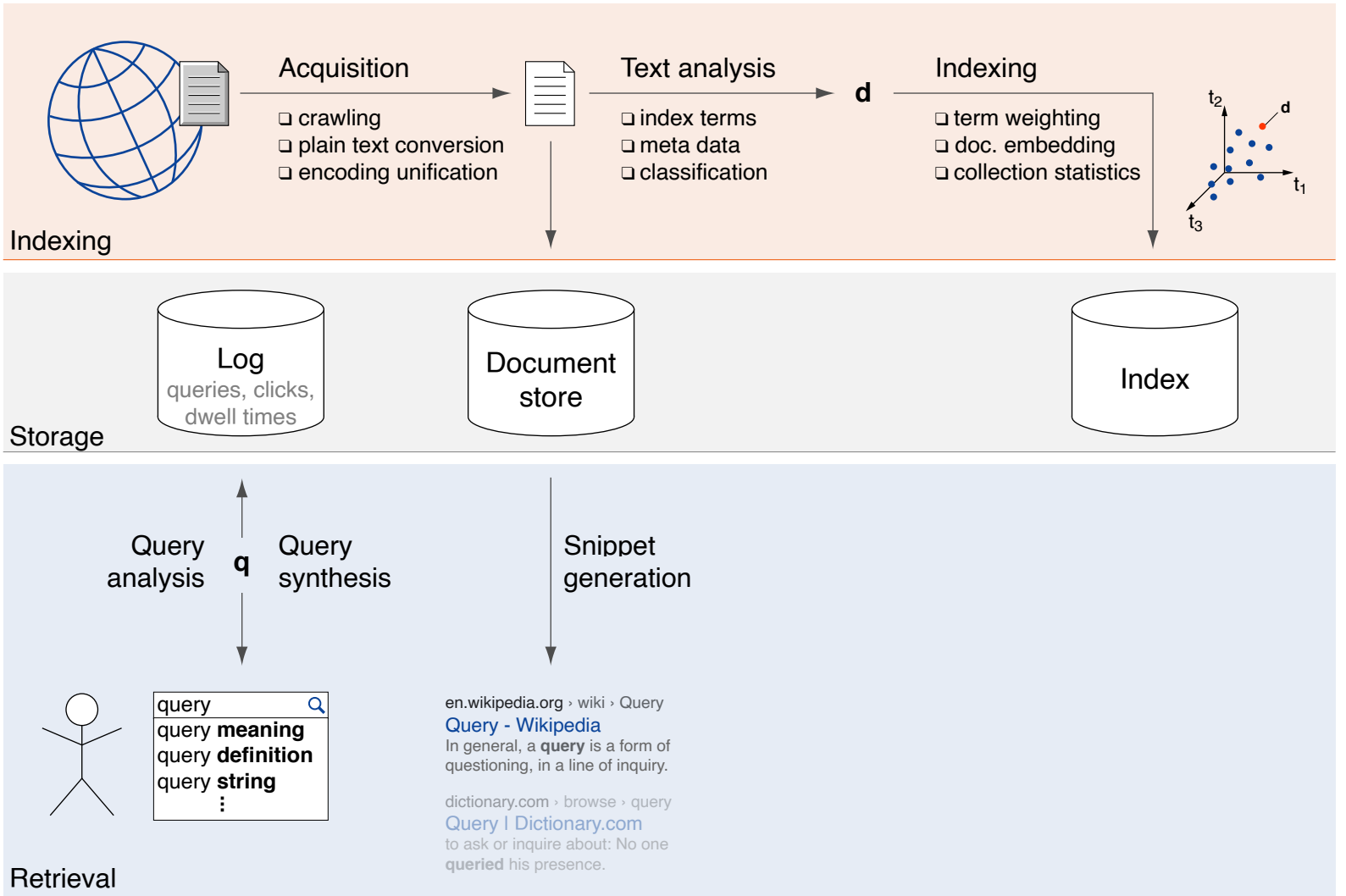
Sources of relevance feedback:

- Direct / explicit relevance feedback
  A user marks retrieved documents as relevant or non-relevant.
- Indirect / implicit relevance feedback
  Relevant documents are identified by analyzing user behavior on the search results page. Relevance signals include: clicked results, dwell times, search abandonment.
- Blind / pseudo-relevance feedback
  The k top-ranked documents are considered relevant without checking.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

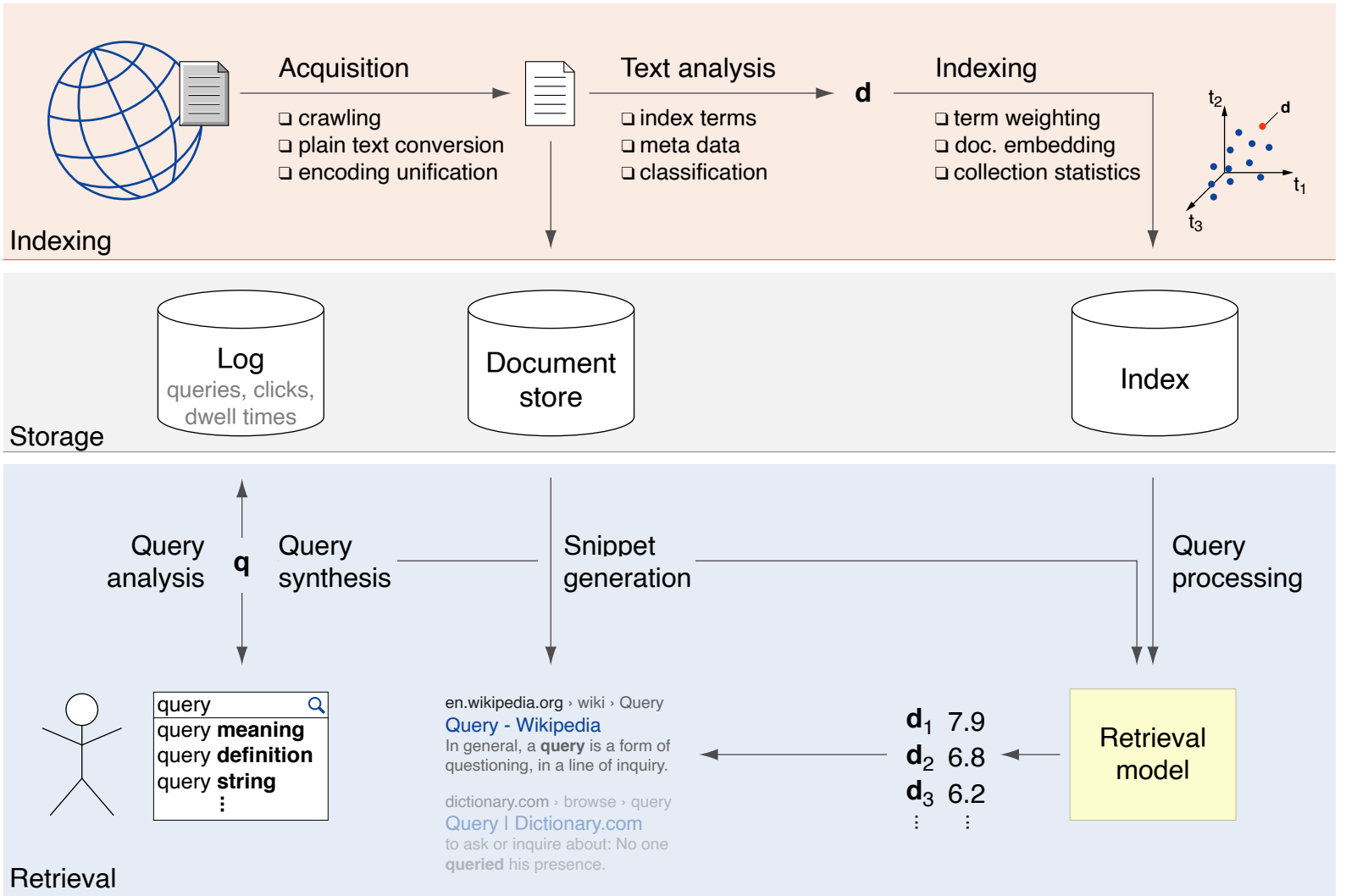
Query Analysis and Synthesis



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Retrieval



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Retrieval

Given a query representation, the ranking step scores and orders the indexed documents with respect to their relevance to the query.

This step marks the keystone of the implementation of the search engine's underlying retrieval model, a theory of how relevance can be quantified.

Retrieval models consist of

- a function to represent documents,
- a function to represent queries, and
- a function to score a document's relevance to a query.

Key components:

- Query Processing
- Relevance Scoring
- Distribution

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Retrieval: Relevance Scoring

Quantification of the relevance of an indexed document d to a query q.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Retrieval: Relevance Scoring

Quantification of the relevance of an indexed document d to a query q.

Let $t \in T$ denote a term t from the terminology T of index terms, and let $\omega_X : T \times X \to \mathbf{R}$ denote a term weighting function, where X may be sets of documents D or queries Q. Then the most basic relevance function ρ is:

$$\rho(q, d) = \sum_{t \in T} \omega_Q(t, q) \cdot \omega_D(t, d),$$

where $\omega_Q(t, q)$ and $\omega_D(t, d)$ are term weights indicating the importance of t for the query $q \in Q$ and the document $d \in D$, respectively.

Why are weights for all terms in T considered?
Why not just the weights of terms actually occuring in query q and document d?

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Retrieval: Relevance Scoring

Quantification of the relevance of an indexed document d to a query q.

Let $t \in T$ denote a term t from the terminology T of index terms, and let $\omega_X : T \times X \to \mathbf{R}$ denote a term weighting function, where X may be sets of documents D or queries Q. Then the most basic relevance function ρ is:

$$\rho(q, d) = \sum_{t \in T} \omega_Q(t, q) \cdot \omega_D(t, d),$$

where $\omega_Q(t, q)$ and $\omega_D(t, d)$ are term weights indicating the importance of t for the query $q \in Q$ and the document $d \in D$, respectively.

Observations:

- A term t may have importance, and hence non-zero weights, for a query q or document d despite not occurring in them. Example: synonyms.
- The majority of terms from T will have insignificant importance to both.
- The term weights $\omega_D(t, d)$ can be pre-computed and indexed.
- The term weights $\omega_Q(t, q)$ must be computed on the fly.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Retrieval: Query Processing

Query processing depends on the index organization strategy:

Document-at-a-time scoring

- Precondition: a total order of documents in the index's postlists is enforced.
  Ordering criterion: document ID or document quality
- Parallel traversal of query term postlists by document ID.
- Each document's score is instantly complete, but the ranking only at the end.
- Concurrent disk IO overhead increases with query length.

Term-at-a-time scoring

- Iterative traversal of query term postlists (e.g., in order of term frequency).
- Temporary query postlist contains candidate documents.
- As relevance scores accumulate, an approximate ranking becomes available.
- More main memory required for maintaining temporary postlist.

Safe and unsafe optimizations exist (e.g., to stop the search early).

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Retrieval: Distribution

The distribution of query processing depends on that of the index.

Query broker / load balancer

- Decides which shard and which replicated copies to access.
- Receives and merges results.

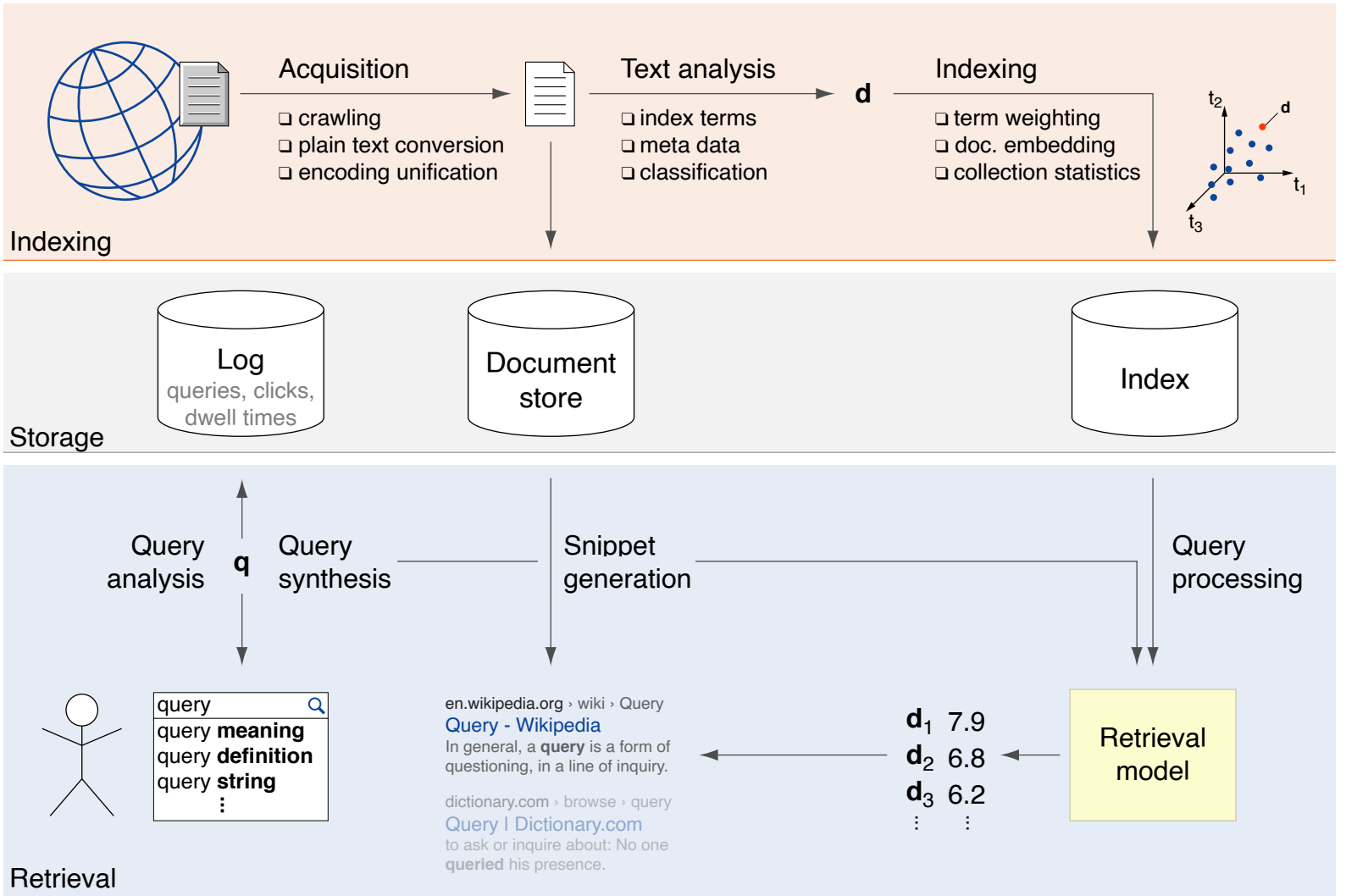
Cache

- Reduces latency by keeping frequently used data close at hand.
  Example: Lookup table in main memory
- Caches may include (smaller) indexes containing important documents only, precomputed search results, temporary postlists, or parts of postlists.
- Caching strategies optimize the usage of the caching hierarchy from operating system to hardware caches.

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

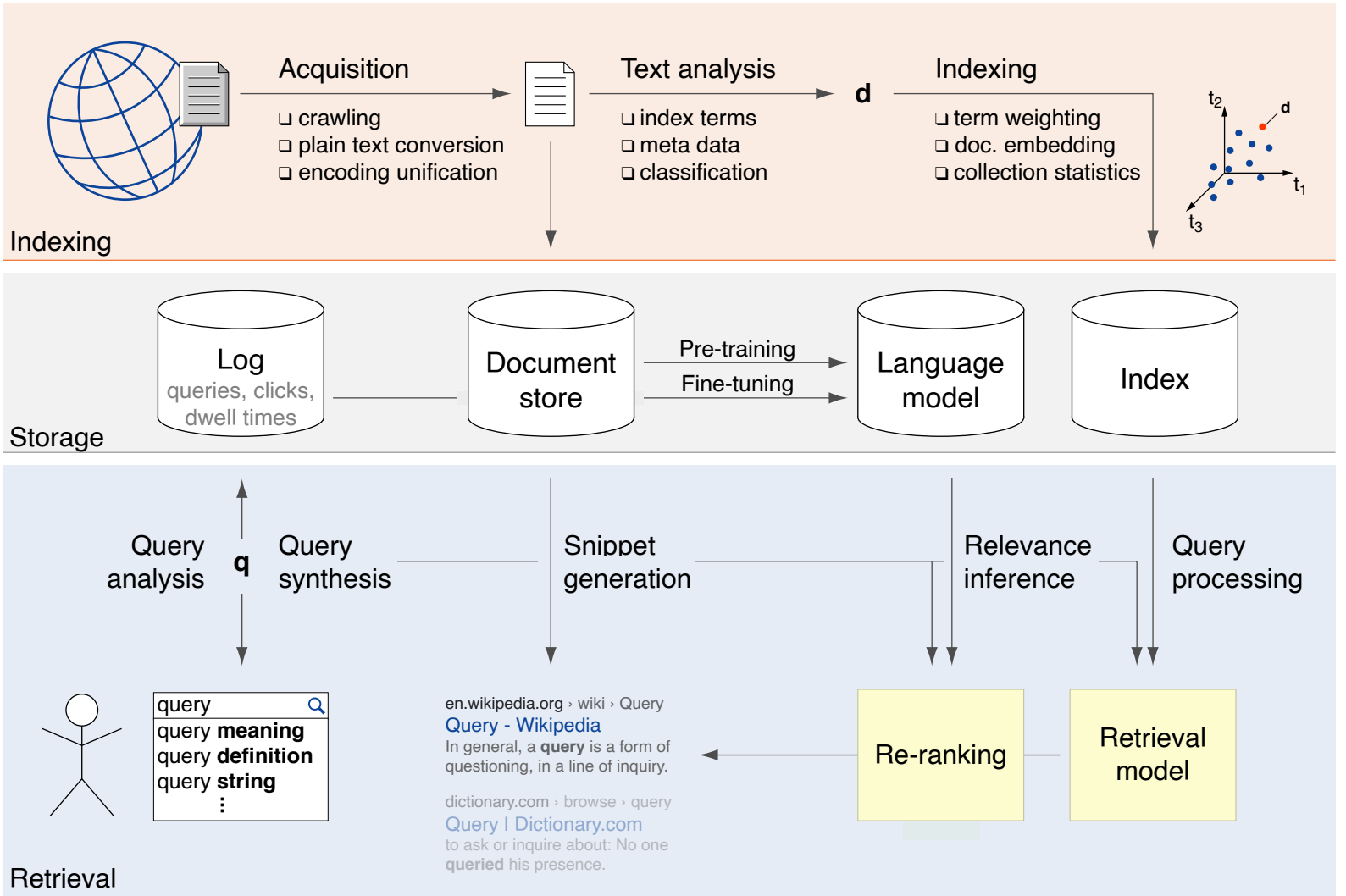
Retrieval



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Re-ranking



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Re-ranking: Learning to Rank (LTR)

Learning objectives:

- Pointwise LTR: Prediction of document d's relevance (score) for query q.
- Pairwise LTR: Prediction whether d is more relevant to q than d'.
- Listwise LTR: Optimization of a ranking effectiveness measure (e.g., nDCG).

Selected LTR Milestones:

- Learning to Rank uses (traditional) ML methods for IR. [Fuhr 1992]
- Early applications suffered from lack of training data. [Manning et al. 2008]
- A breakthrough was achieved using search engine query logs. [Joachims 2002]

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Re-ranking: Learning to Rank (LTR)

State-of-the-art Neural IR models:

- Pointwise NIR: monoT5 [Nogueira et al. 2020]
- Pairwise NIR: duoT5 [Pradeep et al. 2021]
- Listwise NIR: n/a (relevant effectiveness measures are not differentiable)

Selected NIR Milestones:

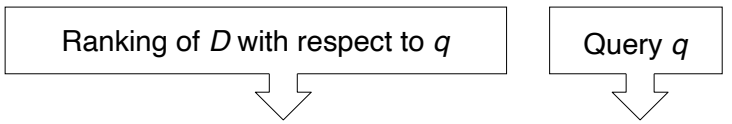
- Language models estimate query likelihood for a document. [Ponte / Croft 1998]
- Early applications suffered from lack of (large pre-)training data. [Lin 2019]
- Transformers, pre-trained at scale severely improves LMs. [Devlin et al. 2018]

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Re-ranking: The Mono-Duo Re-ranking Pipeline

1. BM25 ranking (whole collection)

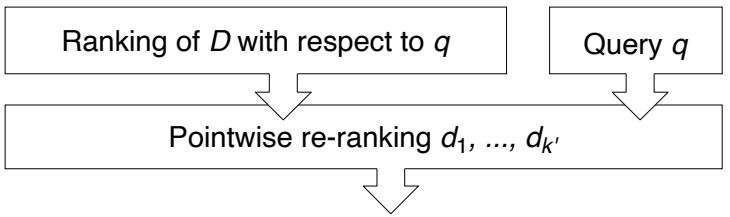


©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Re-ranking: The Mono-Duo Re-ranking Pipeline

1. BM25 ranking (whole collection)
2. Pointwise re-ranking (top-$k' = 1000$)

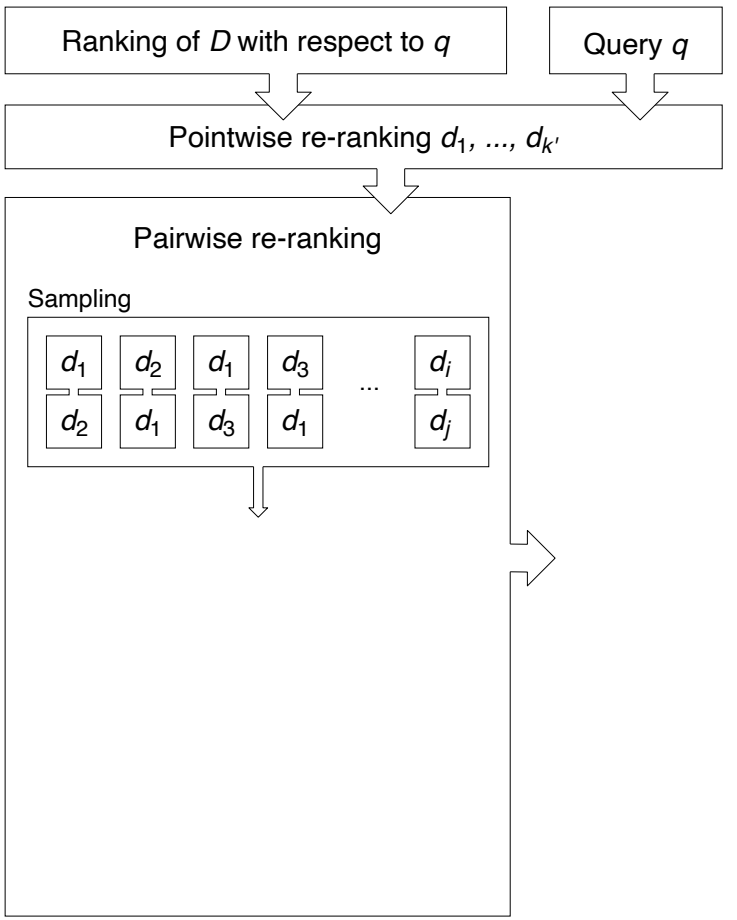


©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Re-ranking: The Mono-Duo Re-ranking Pipeline

1. BM25 ranking (whole collection)
2. Pointwise re-ranking (top-$k' = 1000$)
3. Pairwise re-ranking (top-$k = 50$)
   (a) Sampling pairs for comparison

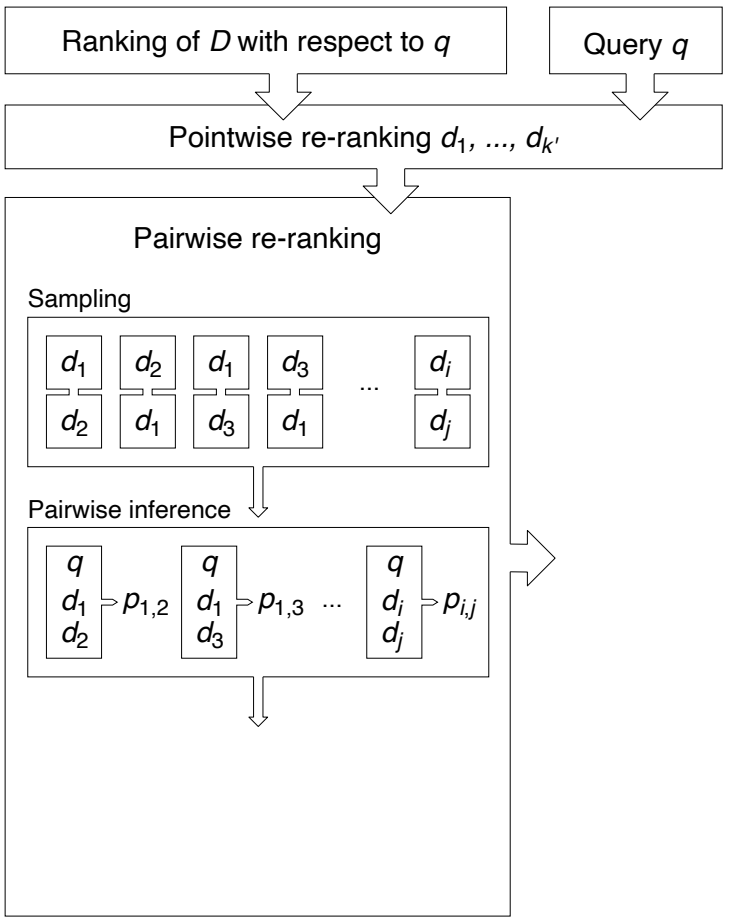


©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Re-ranking: The Mono-Duo Re-ranking Pipeline

1. BM25 ranking (whole collection)
2. Pointwise re-ranking (top-$k' = 1000$)
3. Pairwise re-ranking (top-$k = 50$)
   (a) Sampling pairs for comparison
   (b) Estimating preference probabilities

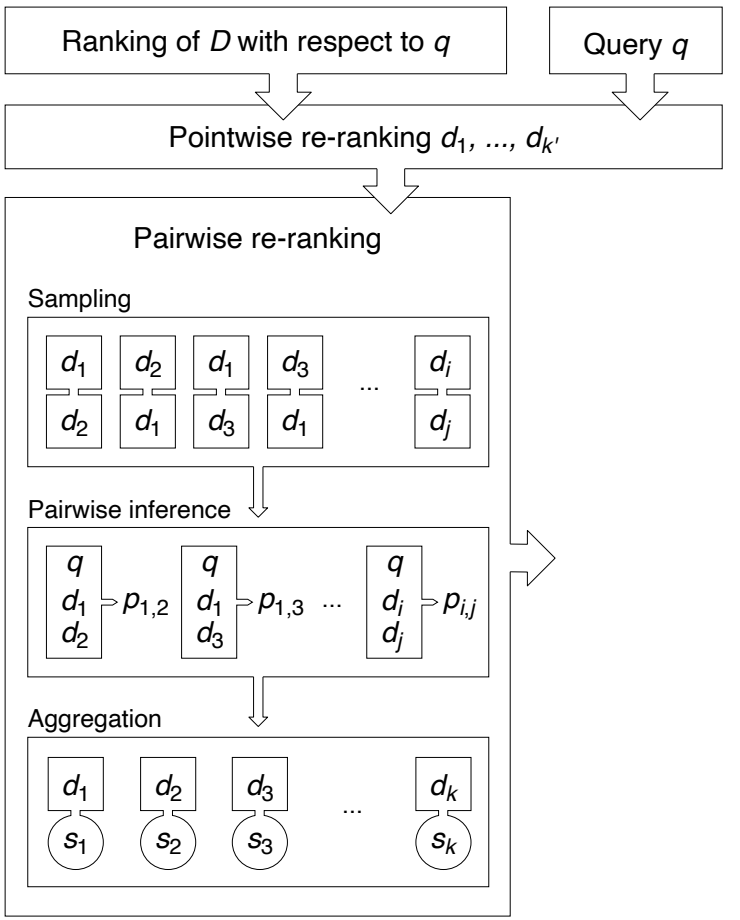


©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Re-ranking: The Mono-Duo Re-ranking Pipeline

1. BM25 ranking (whole collection)
2. Pointwise re-ranking (top-$k' = 1000$)
3. Pairwise re-ranking (top-$k = 50$)
   (a) Sampling pairs for comparison
   (b) Estimating preference probabilities
   (c) Deriving relevance scores

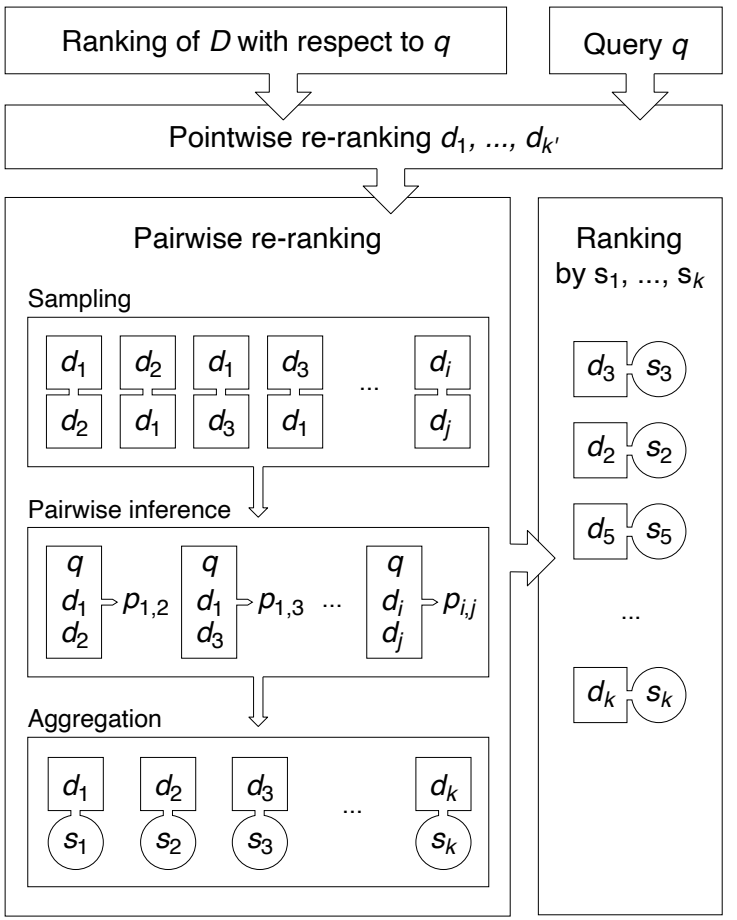


©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Re-ranking: The Mono-Duo Re-ranking Pipeline

1. BM25 ranking (whole collection)
2. Pointwise re-ranking (top-$k' = 1000$)
3. Pairwise re-ranking (top-$k = 50$)
   (a) Sampling pairs for comparison
   (b) Estimating preference probabilities
   (c) Deriving relevance scores
4. Rank by aggregated score



©HAGEN/POTTHAST/STEIN 2023

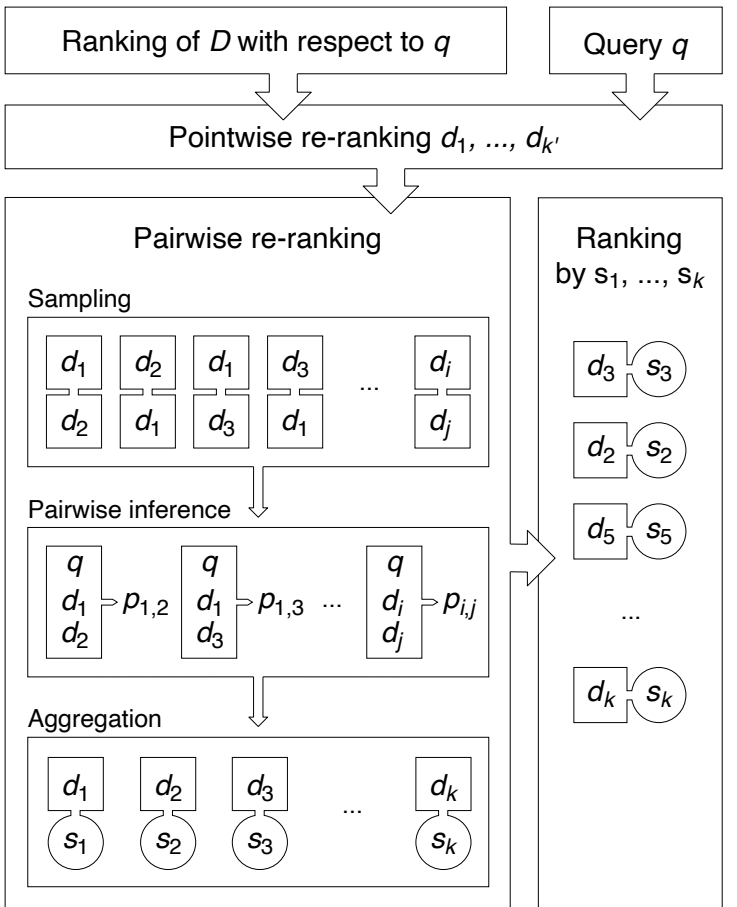
Architecture of a Search Engine

Re-ranking: The Mono-Duo Re-ranking Pipeline

1. BM25 ranking (whole collection)
2. Pointwise re-ranking (top-$k' = 1000$)
3. Pairwise re-ranking (top-$k = 50$)
   (a) Sampling pairs for comparison
   (b) Estimating preference probabilities
   (c) Deriving relevance scores
4. Rank by aggregated score

Model	Depth	Inferences	nDCG@10
BM25	$\|D\|$	none	0.48
monoT5	$k' = 1000$	1000	0.50
duoT5	$k = 50$	2450	0.67

MS MARCO Passage Retrieval



©HAGEN/POTTHAST/STEIN 2023

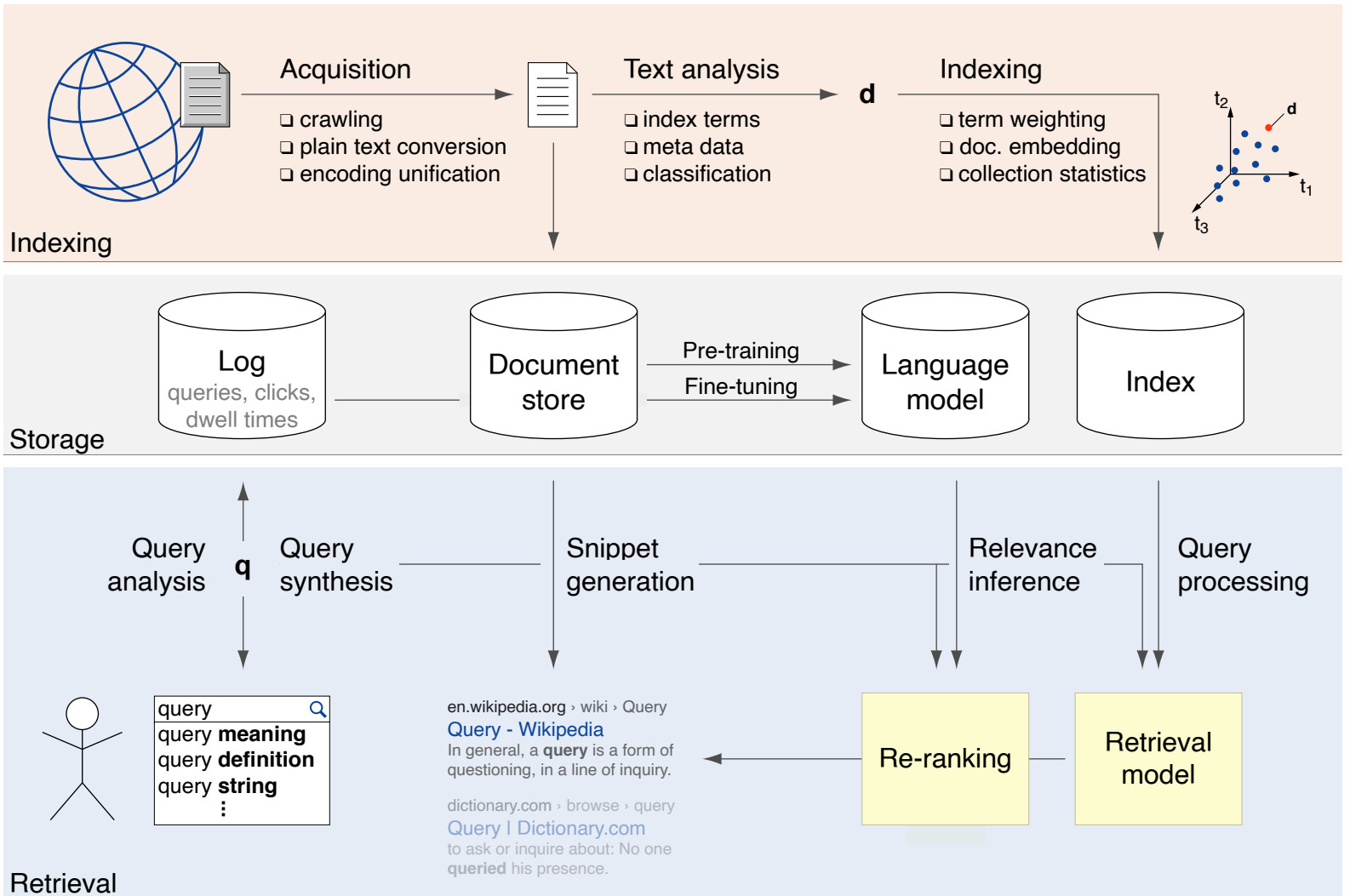
Chapter IR:I

I. Introduction

- Information Retrieval in a Nutshell
- Examples of Retrieval Problems
- Terminology
- Delineation
- Historical Background
- Architecture of a Search Engine
  - Acquisition
  - Text Analysis
  - Indexing
  - User Interface
  - Query Analysis and Synthesis
  - Retrieval
  - Re-ranking
  - Evaluation

©HAGEN/POTTHAST/STEIN 2023

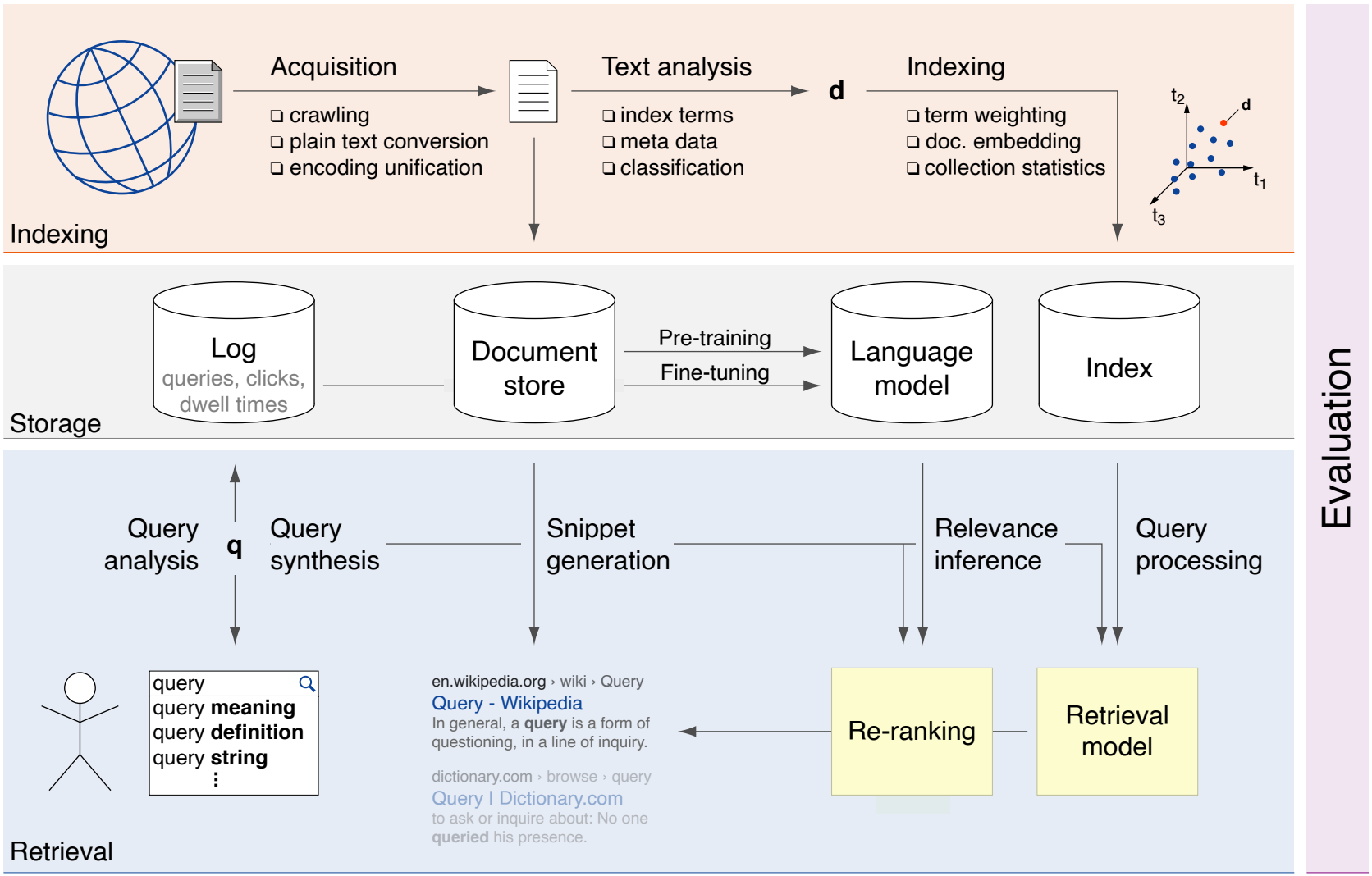
Architecture of a Search Engine



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Evaluation



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Evaluation

Evaluation addresses the analysis of search effectiveness and efficiency.

Retrieval analysis

- Goals: relevant documents first, diversity, novelty, . . .
- (Crowdsourced) acquisition of relevance judgments for query–document pairs
- Measurement theory (e.g., emphasis on top results is common in web search)
- Log analysis of search behavior
- User studies and A/B tests

©HAGEN/POTTHAST/STEIN 2023

User experience analysis

- Goals: usability, user satisfaction, ...
- Log analysis of user behavior
- User studies and A/B testing

Runtime analysis

- Goals: throughput, latency, ...
- Log analysis of system behavior
- Lab experiments and simulation

©HAGEN/POTTHAST/STEIN 2023

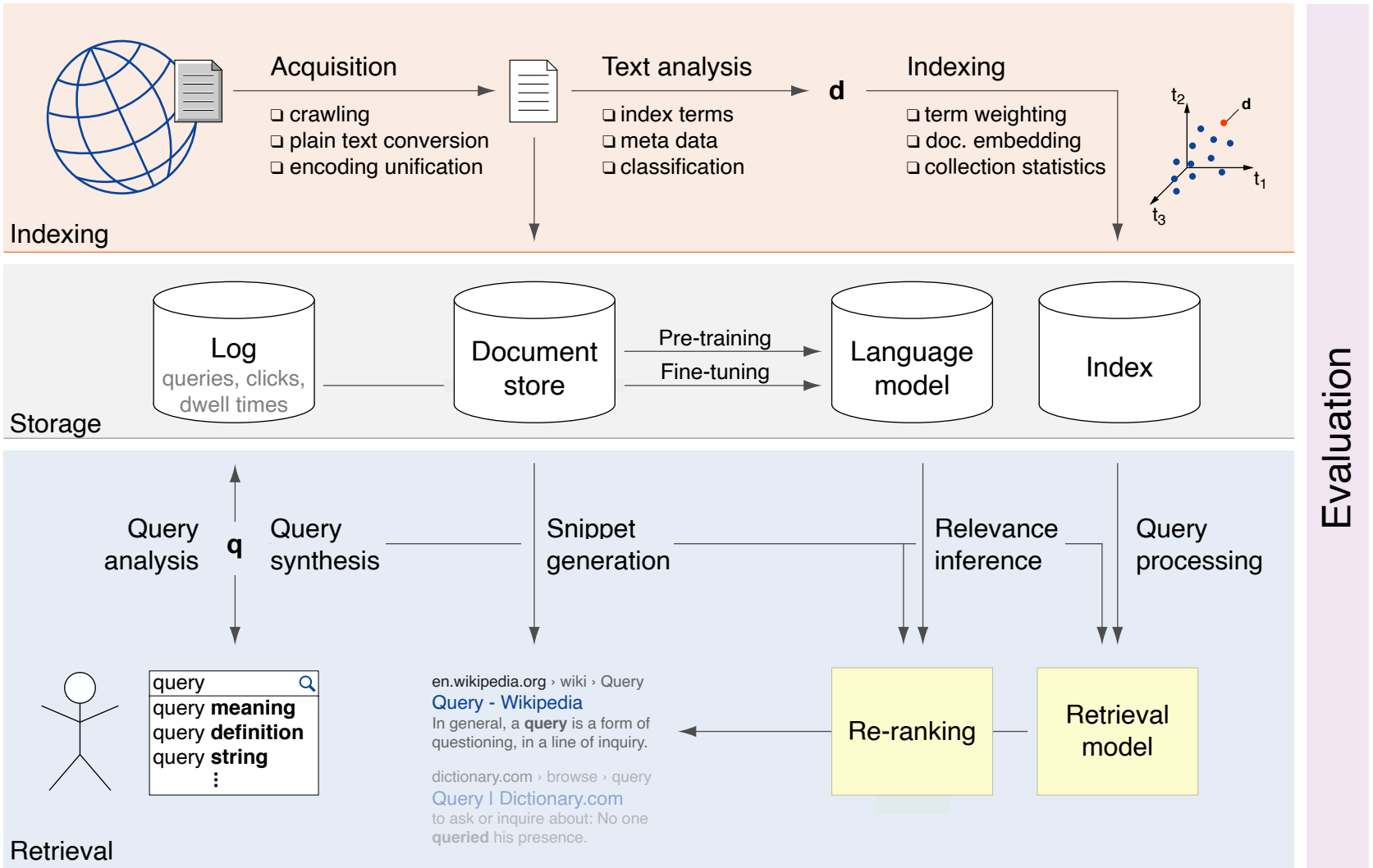
Remarks:

- Evaluation is a systematic determination of a subject's merit, worth, and significance, using criteria governed by a set of standards. It can assist to ascertain the degree of achievement or value in regard to the aim and objectives sought after. The primary purpose of evaluation, in addition to gaining insight into prior or existing initiatives, is to enable reflection and assist in the identification of future change. [Wikipedia]

©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

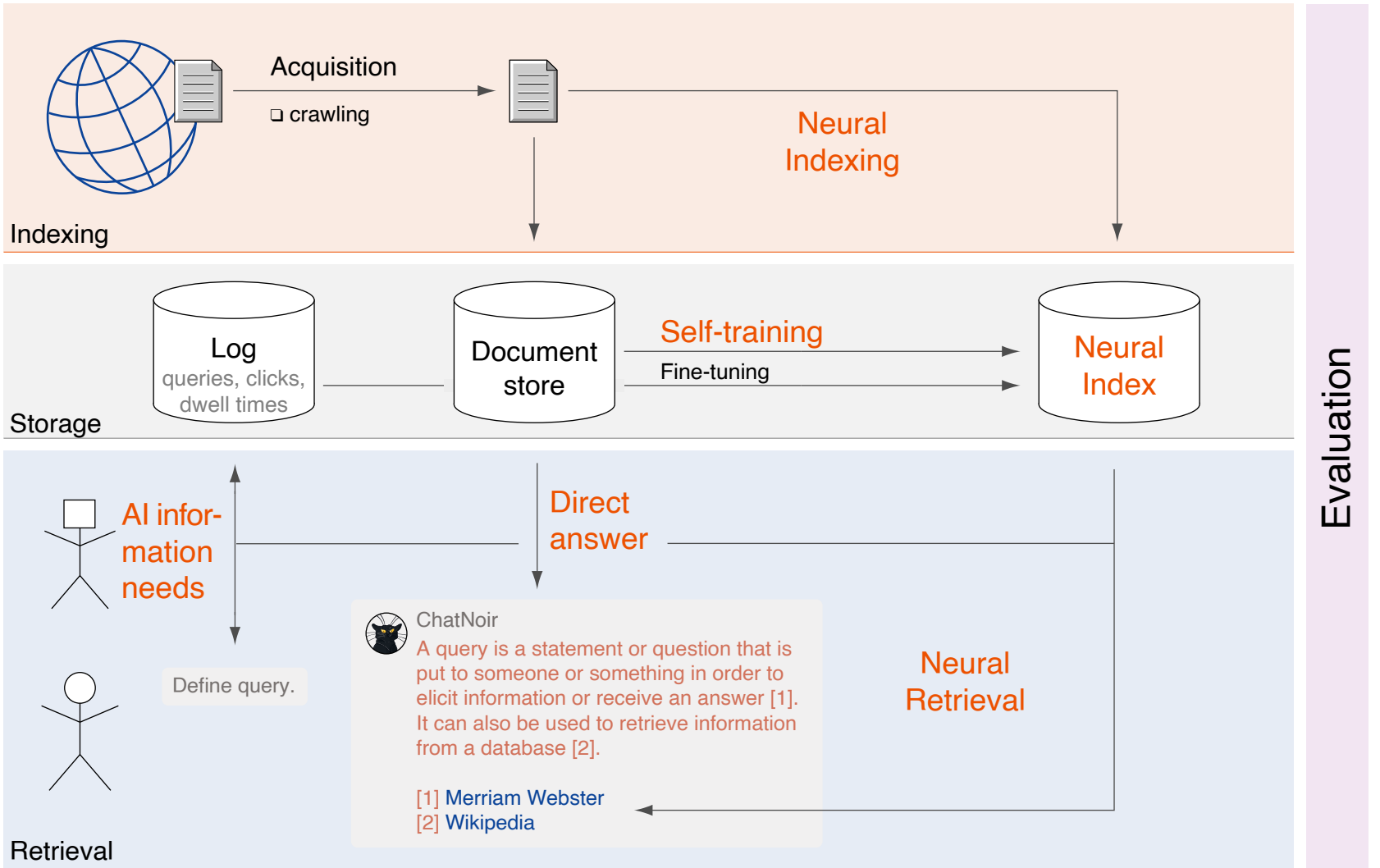
Current Commercial Search Engines



©HAGEN/POTTHAST/STEIN 2023

Architecture of a Search Engine

Neural Indexing and Retrieval



©HAGEN/POTTHAST/STEIN 2023