Chapter IR:II

II. Architecture of a Search Engine

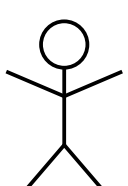
- Overview
- Acquisition
- Text Analysis
- Indexing
- User Interface
- Query Analysis and Synthesis
- Retrieval
- Evaluation

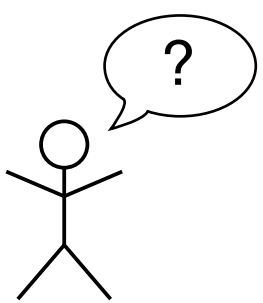
Remarks:

- Software architecture refers to the high level structures of a software system. These structures are needed to reason about the software system. Each structure comprises software elements, relations among them, and properties of both elements and relations.

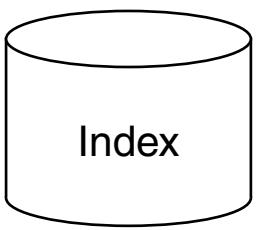
  [Wikipedia]

- 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 software architecture 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]

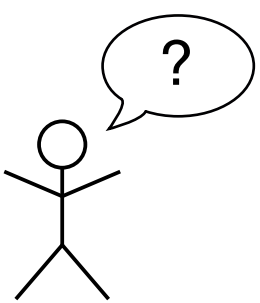




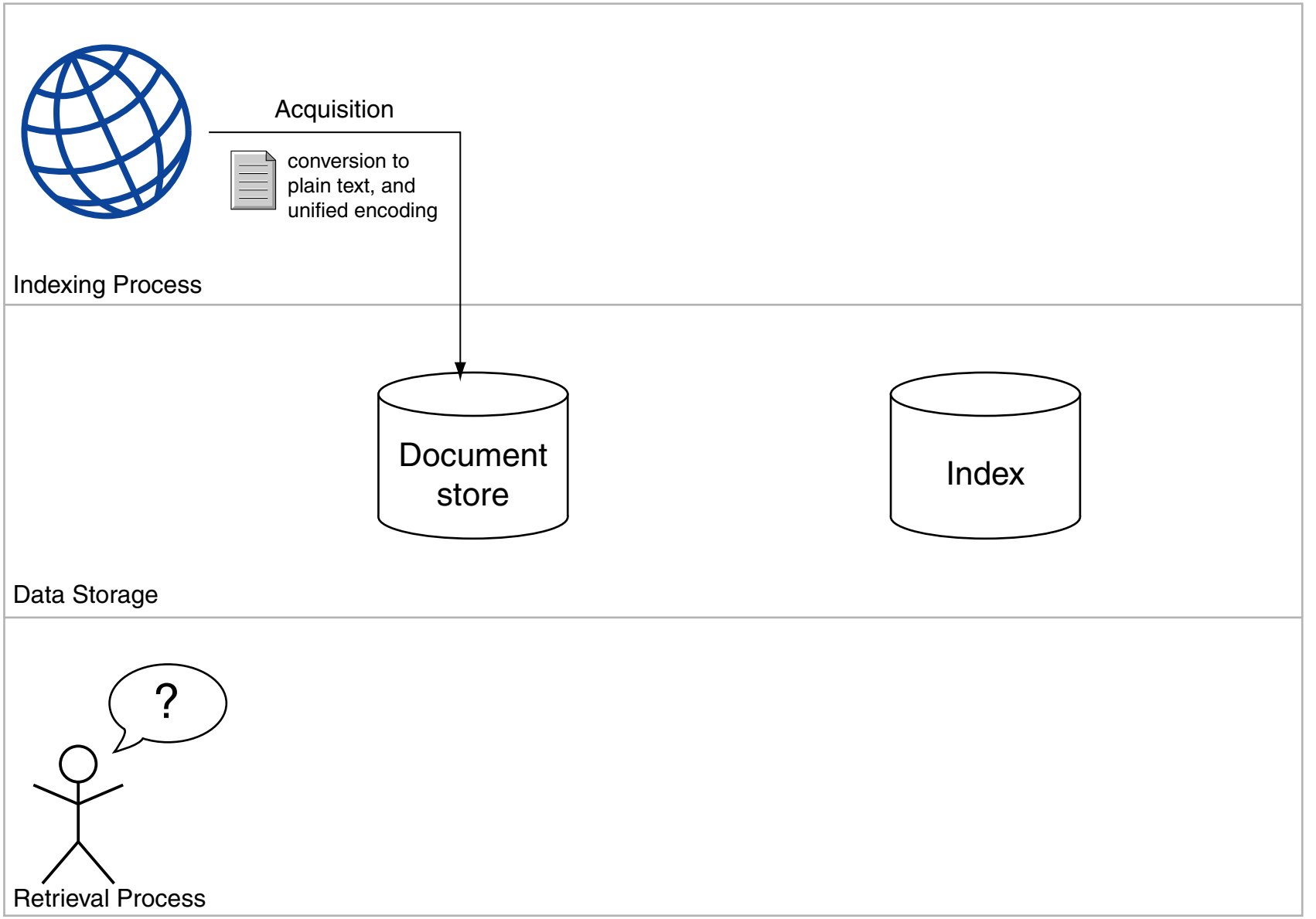
Indexing Process



Data Storage



Retrieval Process



Acquisition

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

Key components:

- Crawler
- Converter
- Document Store

Acquisition

Crawler

A crawler discovers and acquires documents. Several variants are distinguished:

- Web crawler
  - Exploit hyperlinks to discover web pages
  - What are challenges for web crawling?
- Site crawler
  - Web crawler restricted to a site (i.e., domain)
- Focused crawler / topical crawler
  - Acquires documents matching pre-defined topics, genres, or other criteria; discards others while still exploiting their hyperlinks for discovery
  - Requires 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

Acquisition

Crawler

A crawler discovers and acquires documents. Several variants are distinguished:

- Web crawler
  - Exploit hyperlinks to discover web pages
  - Exploration policy (e.g., avoid spider traps), duplicate identification (e.g., URL normalization), parallelization, revisit / update policy, politeness
- Site crawler
  - Web crawler restricted to a site (i.e., domain)
- Focused crawler / topical crawler
  - Acquires documents matching pre-defined topics, genres, or other criteria; discards others while still exploiting their hyperlinks for discovery
  - Requires 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

Remarks:

- Some web sites inform about updates via web feeds (e.g., using RSS or Atom). For such sites, crawling may boil down to subscribing to the feed (i.e., revisiting the feed to learn about updates).

Acquisition

Converter

The converter unifies documents as follows:

- Reformatting / text extraction
  - Documents come in a variety of formats (e.g., HTML, PDF, DOC)
  - Subsequent processing steps require unified input format
  - Extracting plain text from binary documents is lossy (e.g., formatting is lost)
  - Extracting text formatting is important, too, for subsequent steps
  - Using a search engine-specific document format helps (e.g., HTML for web search)
- 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 gets blamed

Acquisition

Document Store

The document store manages all documents acquired:

- Original documents
- Converted documents
  - Why is it necessary 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

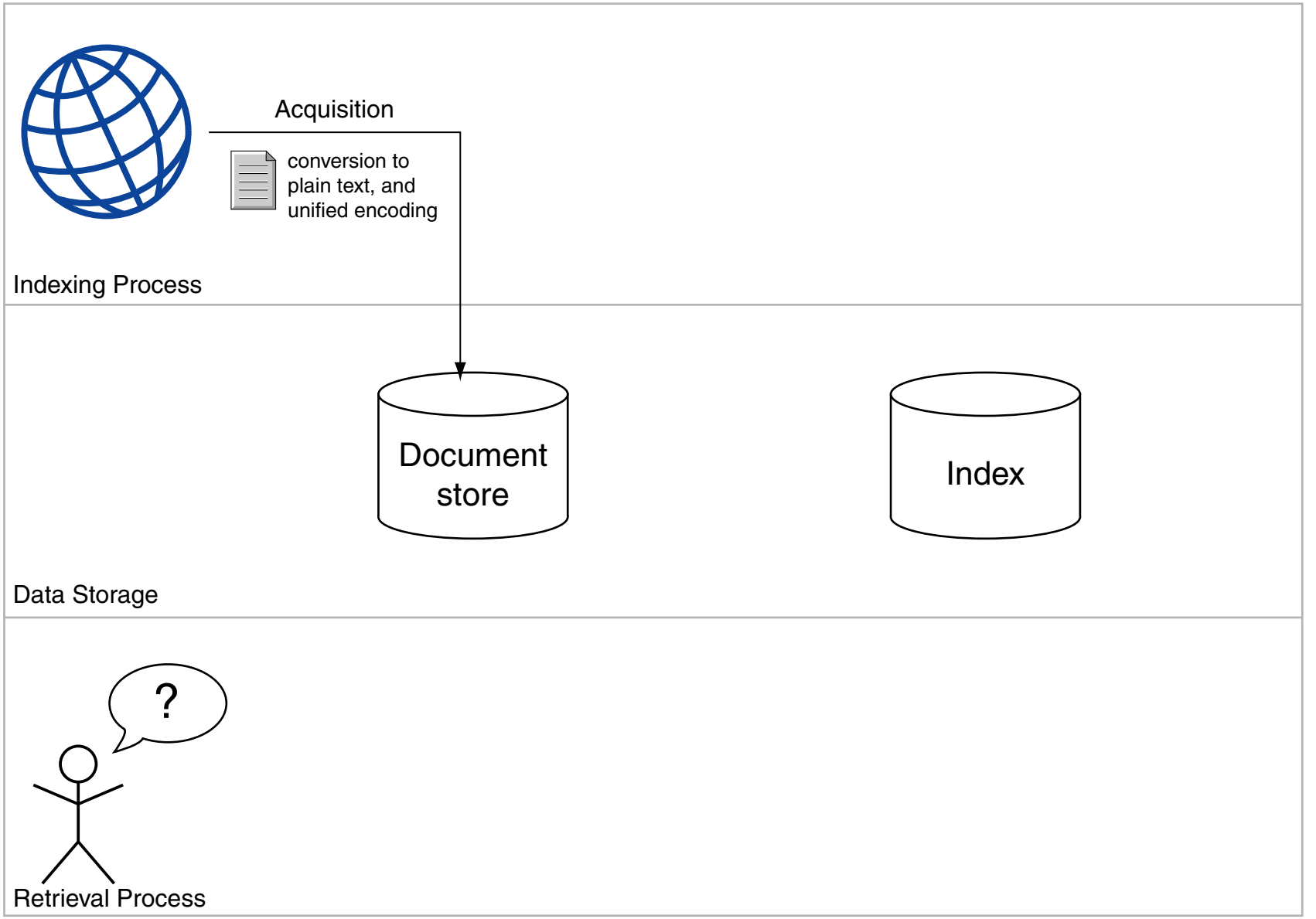
Acquisition

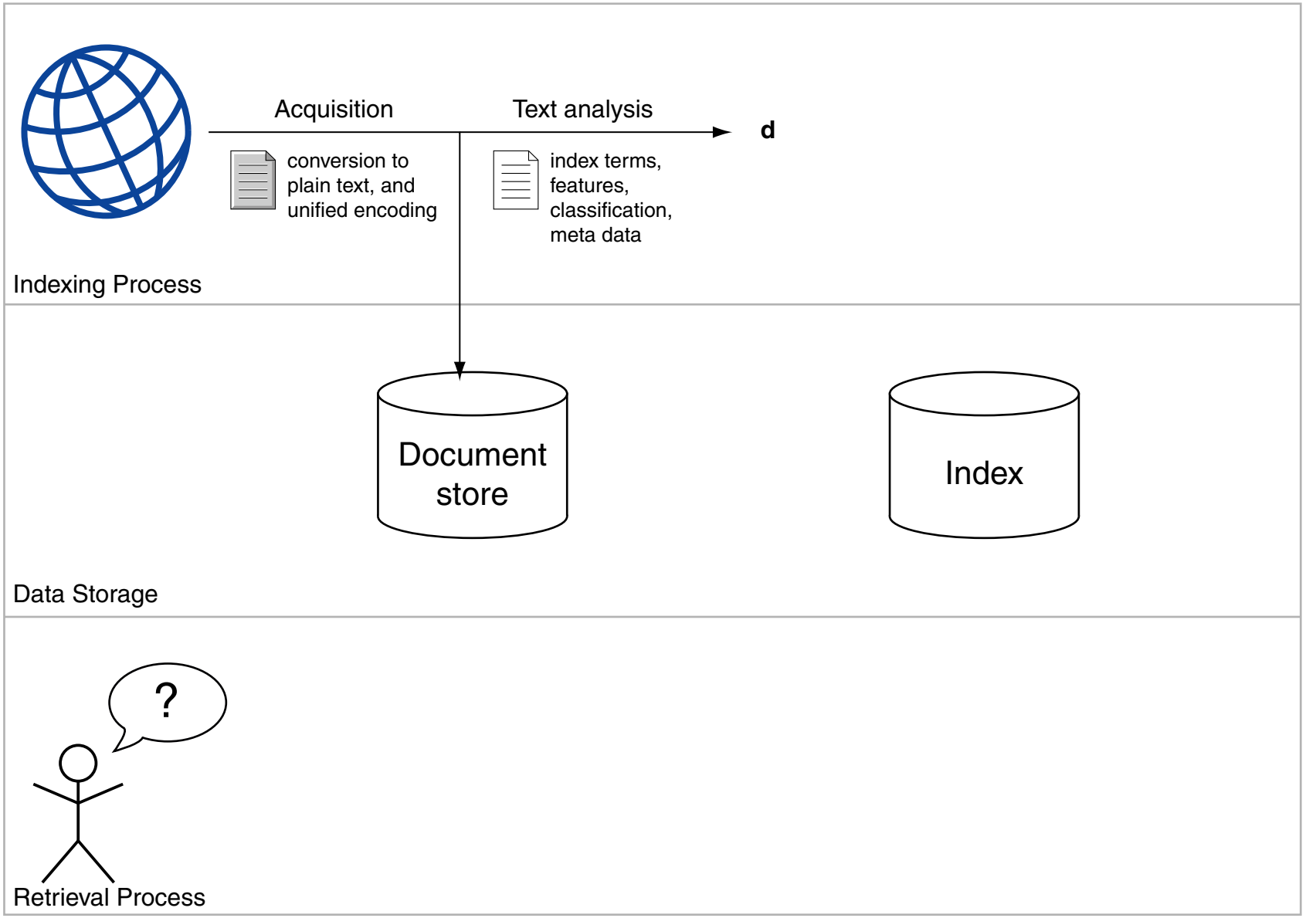
Document Store

The document store manages all documents acquired:

- Original documents
- Converted documents
  - Original may not always be available
  - Fast access for reprocessing (e.g., when processing steps are 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





Text Analysis

The text analysis step extracts from a document's text 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 of 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 an individual measurable property of a document
  - Their purpose is to represent a document for classification
  - Different feature sets are suitable for different classification goals
  - Example classification goals: spam, language, genre, ...
  - Both features and derived class labels may be stored

Text Analysis

The text analysis step extracts from a document's text 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 of 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 an individual measurable property of a document
  - Their purpose is to represent a document for classification
  - Different feature sets are suitable for different classification goals
  - Example classification goals: spam, language, genre, ...
  - Both features and derived class labels may be stored

Key components:

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

Text Analysis

Segmenter

Segmentation means breaking down a document into its constituent parts.
Two levels of segmentation can be distinguished:

- 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.
  - Extraction of text structure and text formatting
  - Often the HTML of a web page is invalid; it must be processed regardless
- Text segmentation
  - Analysis of a web page's plain text with regard to linguistic and structural text units, such as words, sentences, paragraphs
  - Tokenization turns a text string into a sequence of tokens, where a token may be a word or punctuation. Example:
    - Whitespace tokenization: tokens are separated by white space characters
    - Simple definition of a word: token that is an alphanumeric string
    - Why are these definitions insufficient?
  - Lower-cased words are used as index term candidates

Text Analysis

Segmenter

Segmentation means breaking down a document into its constituent parts.
Two levels of segmentation can be distinguished:

- 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.
  - Extraction of text structure and text formatting
  - Often the HTML of a web page is invalid; it must be processed regardless
- Text segmentation
  - Analysis of a web page's plain text with regard to linguistic and structural text units, such as words, sentences, paragraphs
  - Tokenization turns a text string into a sequence of tokens, where a token may be a word or punctuation. Example:
    - Whitespace tokenization: tokens are separated by white space characters
    - Simple definition of a word: token that is an alphanumeric string
    - Punctuation and adjectivization using hyphens not separated by white space
    - Contractions and special characters are neglected
  - Lower-cased words are used as index term candidates

Text Analysis

Stopping

Stopping, also stop word removal, discards a selection of words from the set of index terms of a document, or the entire terminology. 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. Examples: “Wikipedia” appears on every Wikipedia page.
- Domain-specific words
  Words that do not discriminate in a given search domain. Examples: “learning” may be ignored in the education domain, regardless of its frequency.

Upsides of stopping are reduced index size, faster query processing speed, and potential noise reduction in retrieval. Downsides are that cornercases may not be sufficiently covered. Example: search for “to be or not to be”.

Dependent on the retrieval model, stopping may or may not affect retrieval effectiveness. Stopping on text is often more conservative than on queries.

Text Analysis

Stemming / Lemmatization

Stemming aims at reducing 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 at letter 4.
- Lemmatization
  Mapping of a word to its root form, even if it is spelled differently. Example: “saw” and “see”.

The upside of stemming / lemmatization is an increased chance of finding a document when it uses different grammar or derived words different from a query.

What are problems related to aggressive stemming?

Text Analysis

Stemming / Lemmatization

Stemming aims at reducing 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 at letter 4.
- Lemmatization
  Mapping of a word to its root form, even if it is spelled differently. Example: “saw” and “see”.

The upside of stemming / lemmatization is an increased chance of finding a document when it uses different grammar or derived words different from a query.

Downsides are conflations of unrelated words.
Example: “university”, “universe”, and “universal” are stemmed to “univers” by common stemmers. Stemmed words may not be real ones. Lemmatization requires expensive resources.

Effectiveness depends on language (e.g., English vs. Arabic).
Alternative approach: query expansion.

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 this 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. Hyperlinks often enclose text, and the text before and after hyperlinks may justify the link. These anchor texts are added to the text extracted from the linked page to be indexed.

Text Analysis

Information Extraction

Information Extraction aims at identifying more complex index terms by means of natural language processing technology (computationally expensive):

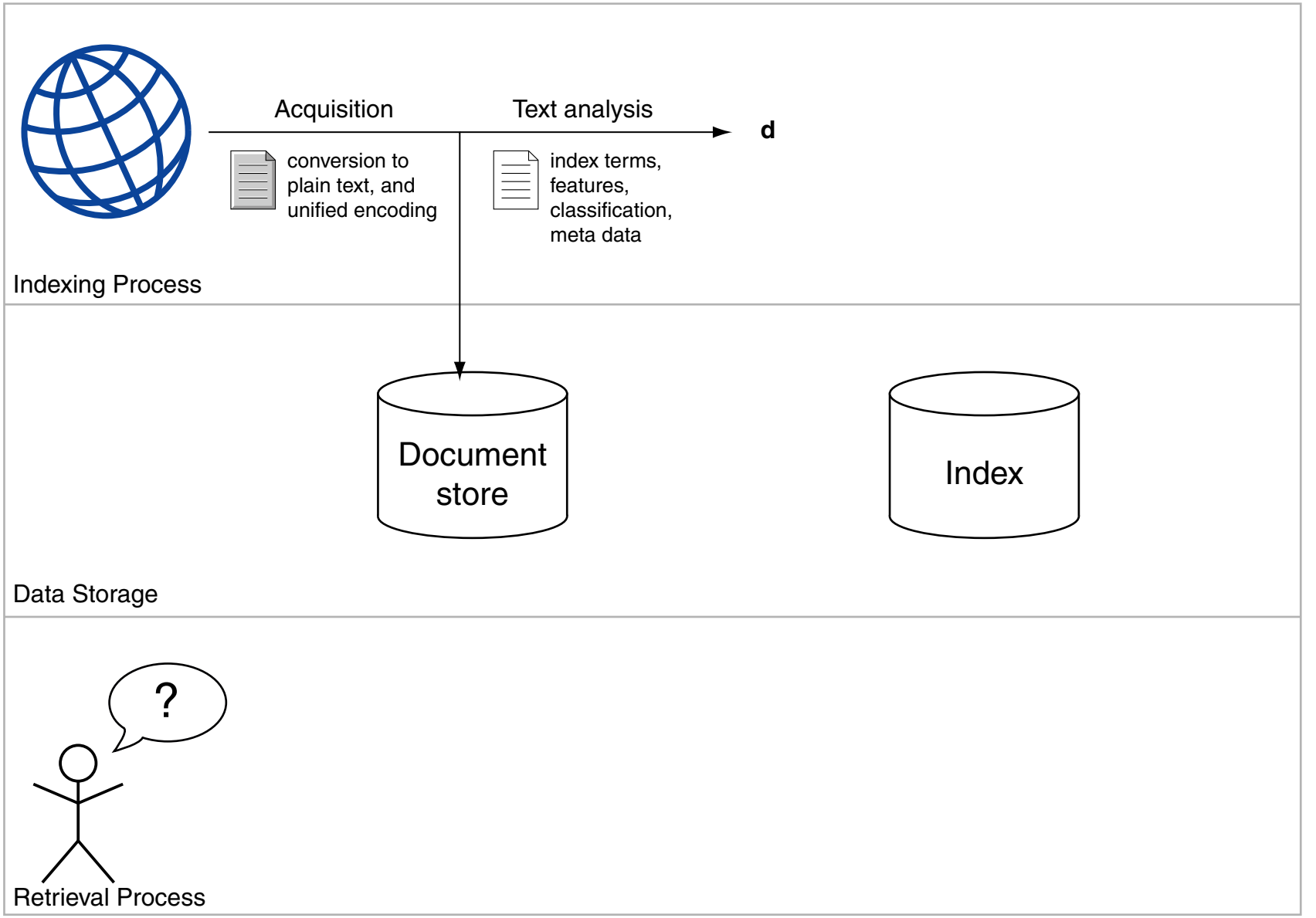
- Noun phrases
  Phrases which have a noun as its head word (i.e., a noun and any word that modifies it). Examples: "*The yellow house* is for sale.", "I want *a skateboard*."
- Named entities
  Words or phrases that designate something in the "real" world (e.g., a place, a person, an organization, etc.).
- 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, one of the most difficult problems of natural language processing.
- Relation detection
  Extraction of relations between named entities mentioned in the text. Example: "*Bill* lives in the *USA*."
- Semi-structured information extraction
  Extraction of tables, quotes, references, comments, etc.

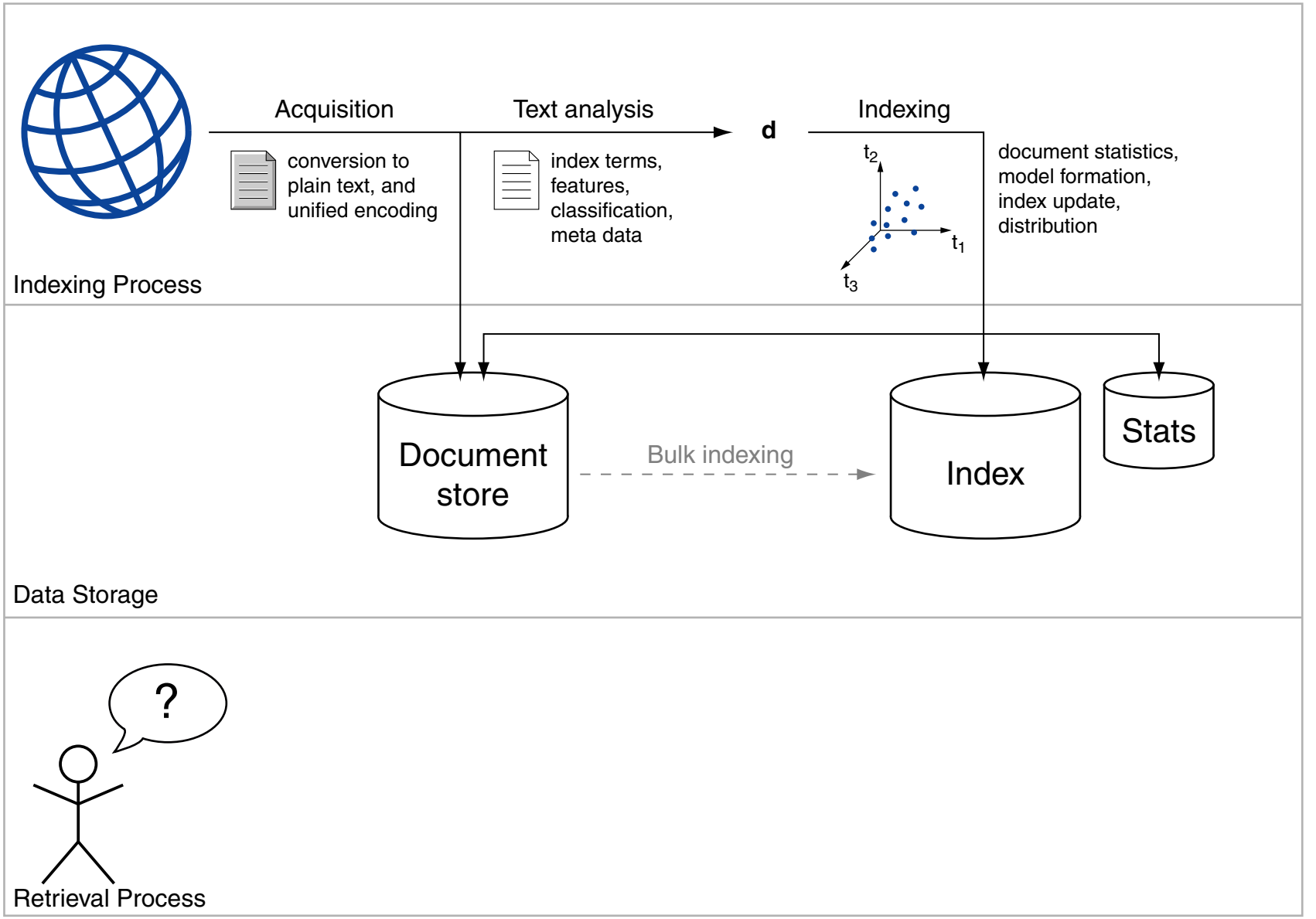
Text Analysis

Classification

Classification applies machine learning to identify specific types and kinds of documents as well as categorizing them. It is based on the features extracted during text analysis. Common classification goals are:

- Spam detection / malware detection
  Determines whether a website / web page tries to subvert the search engine's ranking (spam), or to harm its users (malware).
- Language identification
  Determines the (main) language of a web page.
- Topic categorization / cluster analysis
  Determines the topic of a document, either by assigning pre-defined subject descriptors (e.g., sports, politics, technology, etc.), covering topics important to users or in a search domain, or by identifying topics using cluster analysis, where identified clusters must be labeled. Topics may overlap and form hierarchies.
- Genre categorization
  Determines the genre of a web page (e.g., personal home page, message board, blog, shop, etc.). There is no commonly agreed list of web genres. Genres overlap and form hierarchies, and may depend on the search domain.





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. Other data structures include the suffix array and the signature file.

Key components:

- Document Statistics
- Term Weighting
- Index Construction
- Distribution

Indexing

Document Statistics

Gather and store auxiliary information about documents, required for fast online query processing (i.e., to score and rank documents when a query is submitted). Such information may include:

- Term frequencies per document, topic, and genre
- Document frequencies per term
- Term positions per document
- Document lengths

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

Indexing

Term Weighting

For each index term of a document, calculate a weight indicating its importance with respect to the document, allowing for document scoring with respect to a query. Common term weighting schemes:

- Term frequency (*tf*)
  Logarithm of the 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 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 performance.

Pre-computing the term weights and storing them in the index or auxiliary data structures speeds up document scoring. Other weighting schemes are calculated based on information about the query.

Indexing

Index Construction

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

- Bulk indexing
  Creates an index by processing all transformed documents. This happens offline; once ready, the currently used index is replaced with the new one.
- Index update
  Updates an existing index with new documents as they appear. This happens online, while the index is in use.

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	$\rightarrow$	$d_1, w_{5,1}$			
$\vdots$					

Indexing

Distribution

The distribution of an index (also called sharding or partitioning) across machines and data centers facilitates parallel usage.

- 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?

Indexing

Distribution

The distribution of an index (also called sharding or partitioning) across machines and data centers facilitates parallel usage.

- Document distribution
  - Split collection; smaller indexes for sub-collections on different machines
  - Enables parallelism for indexing and query processing
  - Smaller indexes often faster due to caching effects
- 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

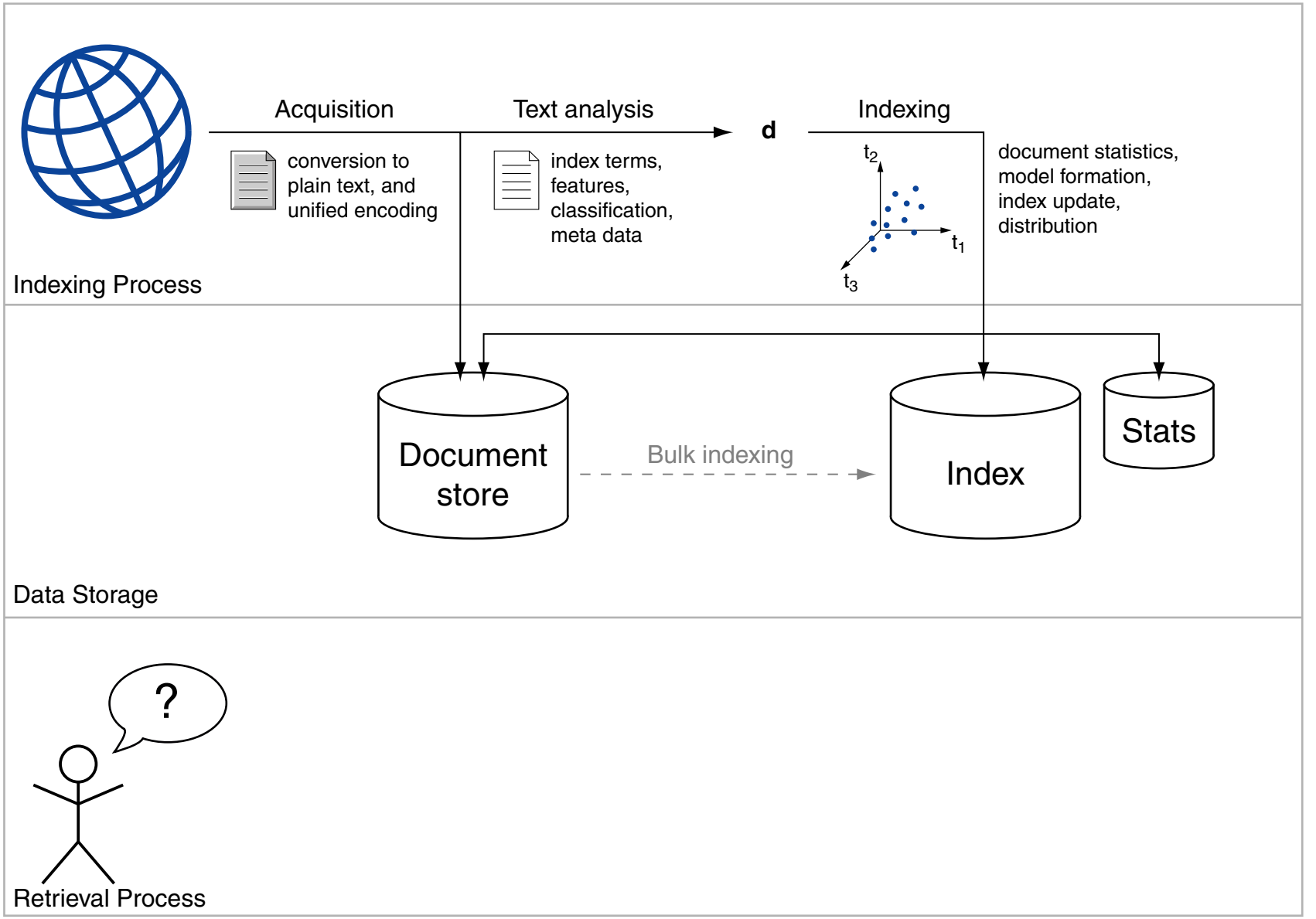
Chapter IR:II

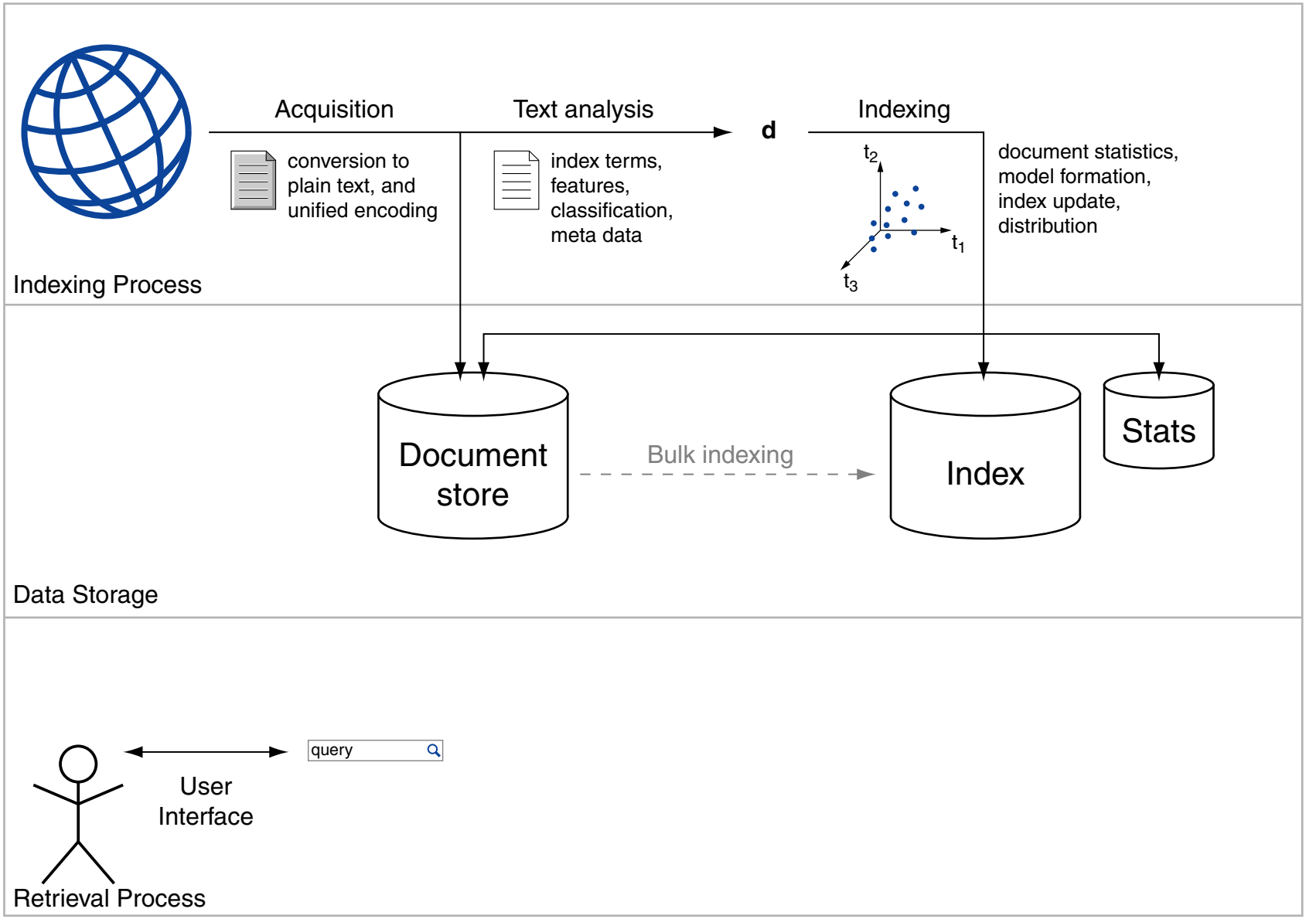
II. Architecture of a Search Engine

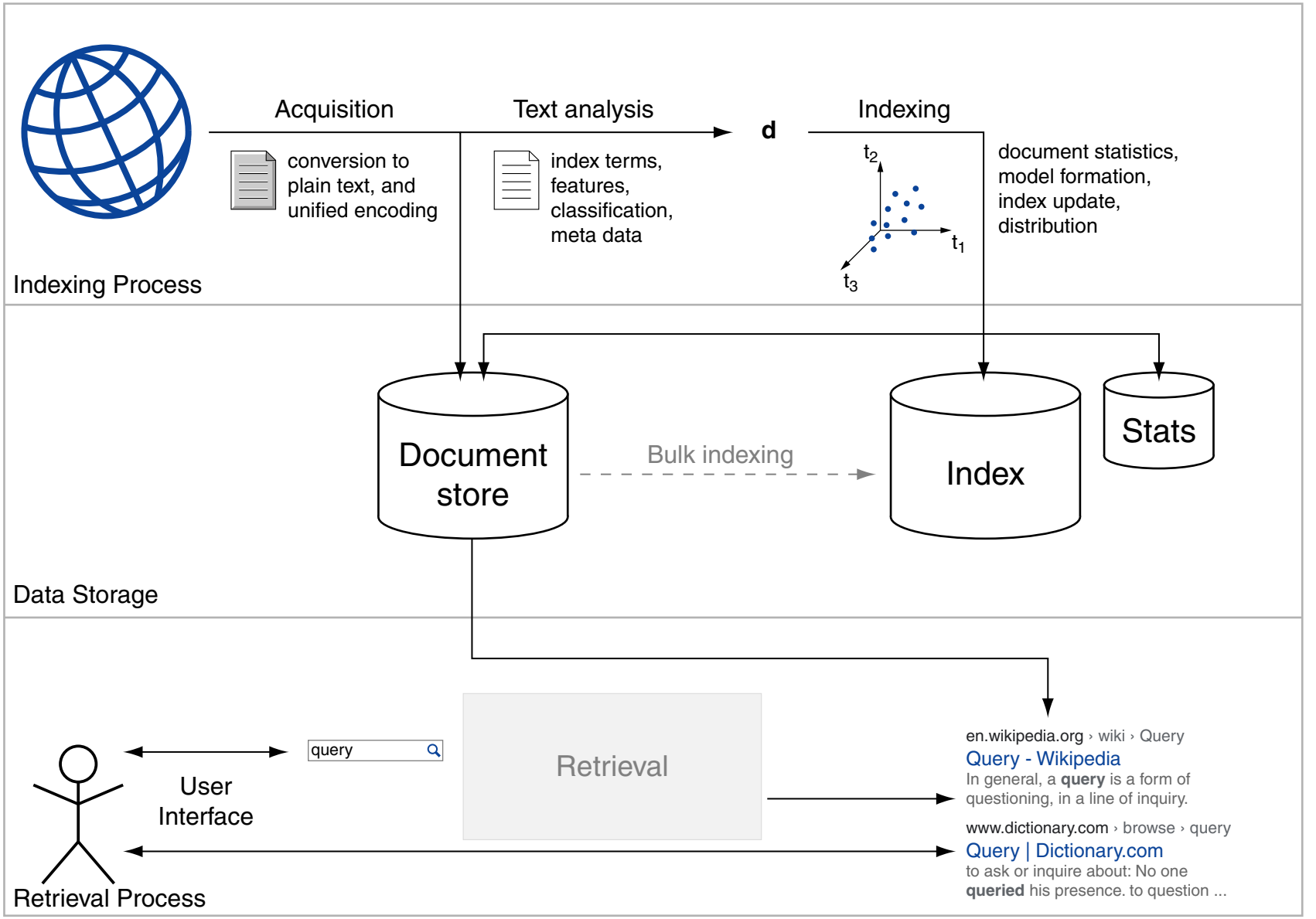
- Overview
- Acquisition
- Text Analysis
- Indexing

- User Interface
- Query Analysis and Synthesis
- Retrieval

- Evaluation







User Interface

A search engine's user interface enables user interaction with the system. They are designed with specific usage scenarios in mind.

Basic user interface:

- Query submission
- Result presentation

User Interface

A search engine's user interface enables user interaction with the system. They are designed with specific usage scenarios in mind.

Advanced user interface:

- Query refinement
- Result exploration
- Session support

User Interface

A search engine's user interface enables user interaction with the system. They are designed with specific usage scenarios in mind.

Advanced user interface:

- Query refinement
- Result exploration
- Session support

Key components:

- Query Language
- Result Presentation

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 –))
- What other operators do you know?

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 (*, ~100)
- 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], so that web search engines cannot expect users being experts of the query language.

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

User Interface

Result Presentation

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

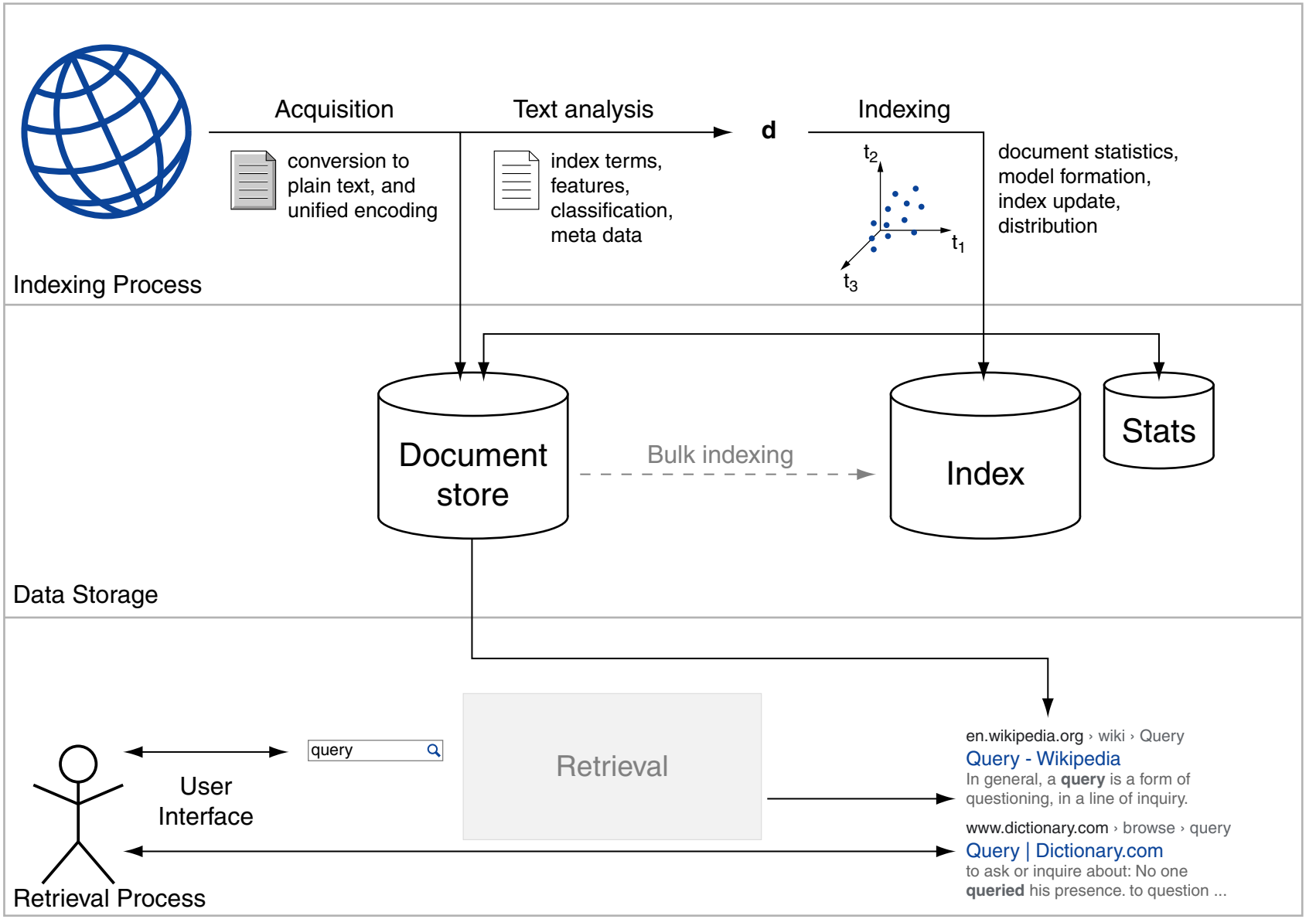
- Snippet generation
  Accesses the original web page and extracts sentences and phrases that summarize it, dependent on the query. The query's terms are highlighted in the snippet, including alternative inflections or synonyms.

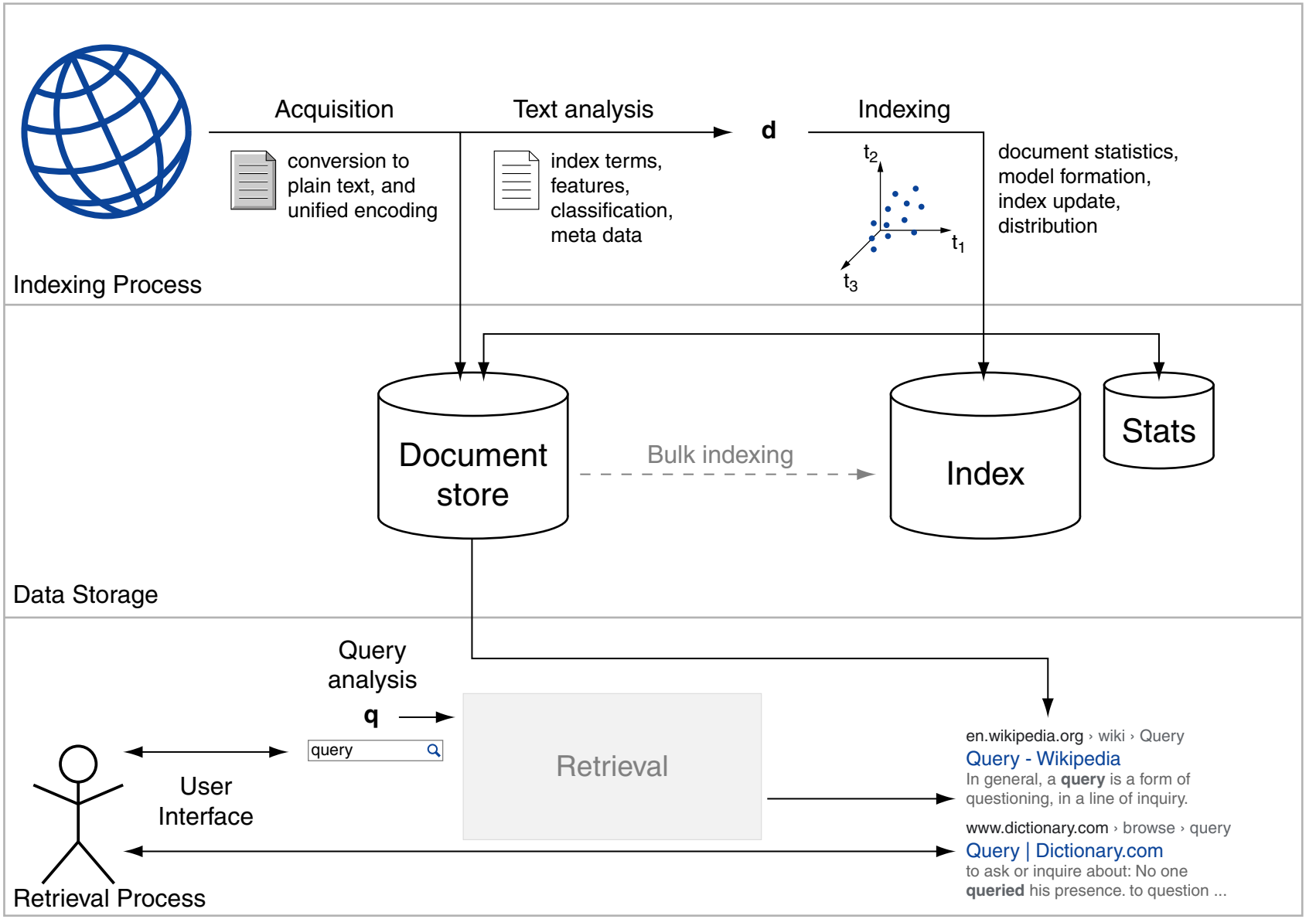
  en.wikipedia.org › wiki › Query
  Query - Wikipedia
  In general, a **query** is a form of questioning, in a line of inquiry.

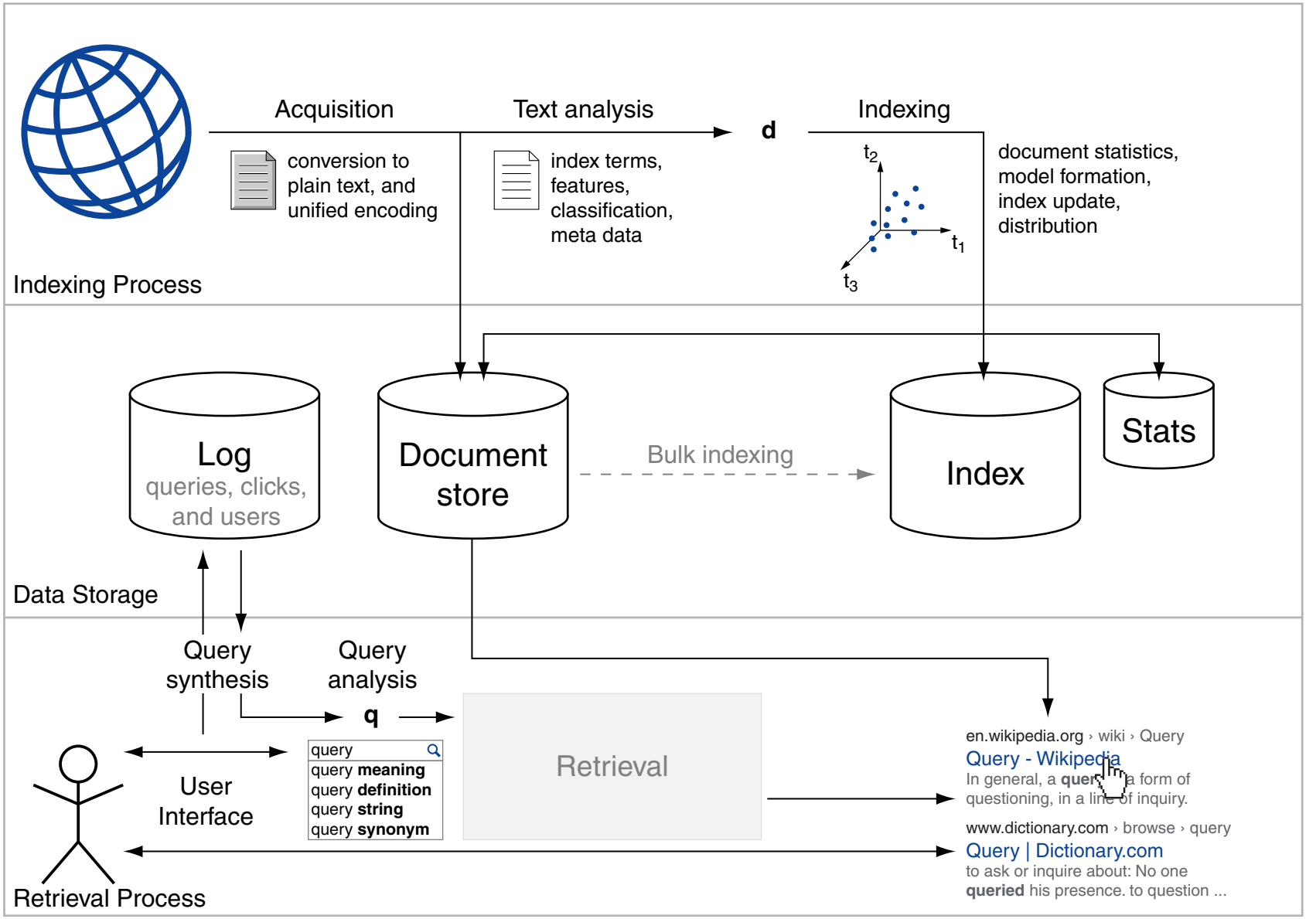
- Universal search (e.g., oneboxes)
  Determines whether other specialized search engines can supply relevant results. Ranks the oneboxes into the search results as per their importance compared to the organic web results.

- Ad retrieval
  Accesses another search engine specifically tailored to the retrieval of ads relevant to a query from all ads offered by advertisement partners. This runs in parallel to the retrieval of organic search results.

- Facets / Categorization
  If meta data about documents is available, a side bar with that allows to set constraints about the meta data can be displayed. Alternatively, documents can be categorized using cluster analysis and cluster labeling.







Query Analysis and Synthesis

Text Analysis

Query analysis maps a query's keywords to index terms to enable successful retrieval, using 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.

Query analysis is also often called query understanding.

Query Analysis and Synthesis

Logging

A search engines keeps of logs of the following user interactions:

- Queries
  Every query submitted 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.

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 Ms. 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]

Query Analysis and Synthesis

Query Rewriting

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

Replacement of terms:

- Spelling correction
  Detection and correction of misspelled terms in a query. Spell correctors rely on language modeling to determine the probability of a query, and an error model to determine the probability of a misspelling given an intended query.
- Query suggestion
  Feedback about the query at various degrees of urgency, ranging from small hints ("Did you mean ...") to automatic replacement, dependent on confidence in the suggested alternatives. The query log allows for identifying similar queries and for suggesting completions.

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.

Query Analysis and Synthesis

Query Rewriting (continued)

Query rewriting changes the query to improve it's 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.

Specialization:

- Query segmentation
  Identification of well-known phrases and multi-term concepts in a query, enclosing them in quotes to ensure their occurrence in retrieved documents.
- Query scoping
  Focusing of (parts of) a query on specific fields of a document, e.g., the title, or the body of a document.
- Personalization
  User profiles allow for tailoring search results to the user's context and interests.

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." → "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 from based on query context.

Synonyms, hypernyms, hyponyms:

- Dictionary-based
  Add synonyms of query terms found in a dictionary or thesaurus.
  Problems: exactness, diversity (e.g., "computer" → "laptop", "web" → "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.

Query Analysis and Synthesis

Query Expansion: Relevance Feedback

Relevance feedback refines a query in multiple retrieval runs:

1. Retrieval of documents using the current query.
2. Identification of relevant / irrelevant documents among the top-ranked ones.
3. Extraction of terms related to the user's information need from the relevant documents.
4. Addition the extracted terms to the query; next retrieval run.

Query Analysis and Synthesis

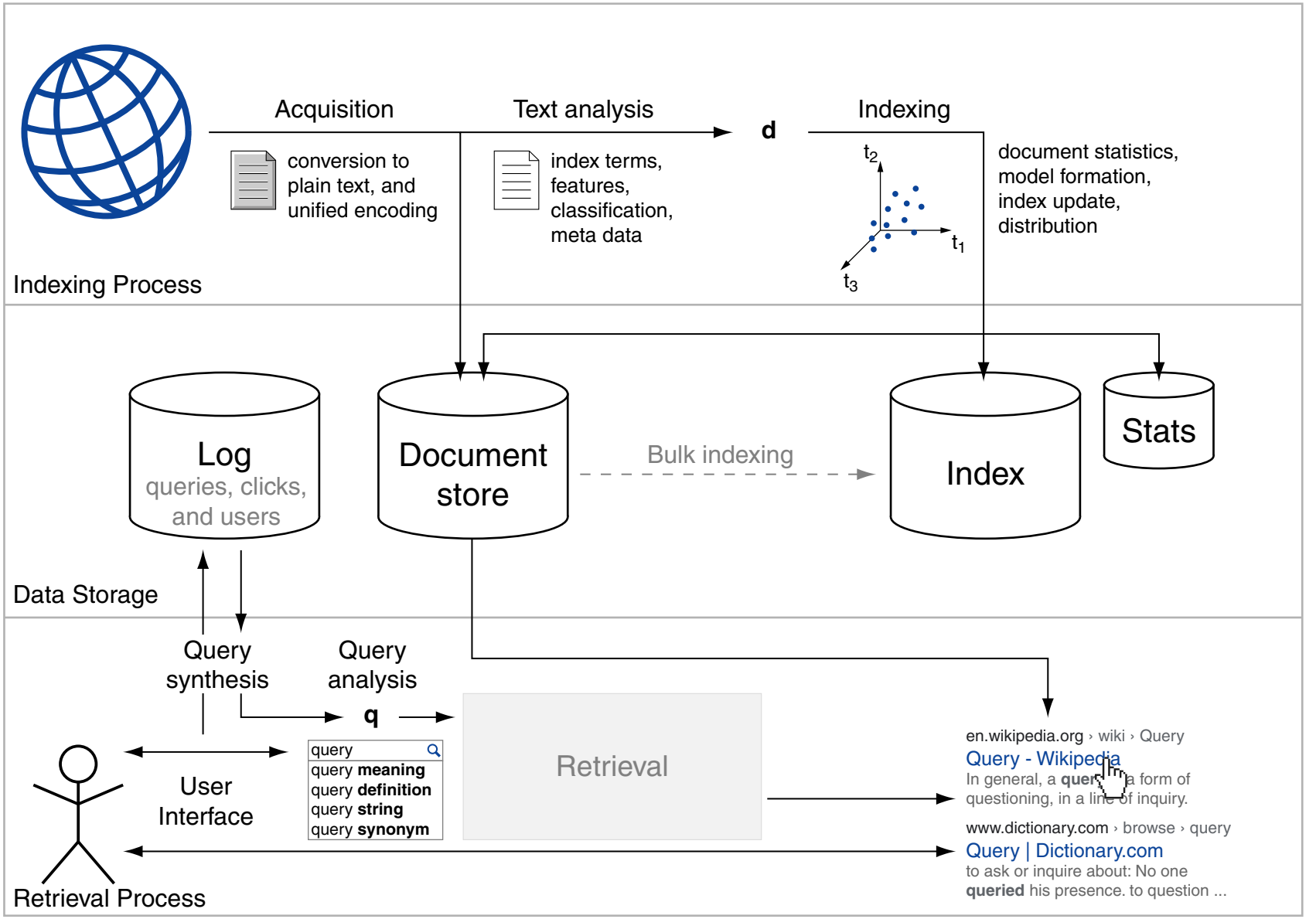
Query Expansion: Relevance Feedback

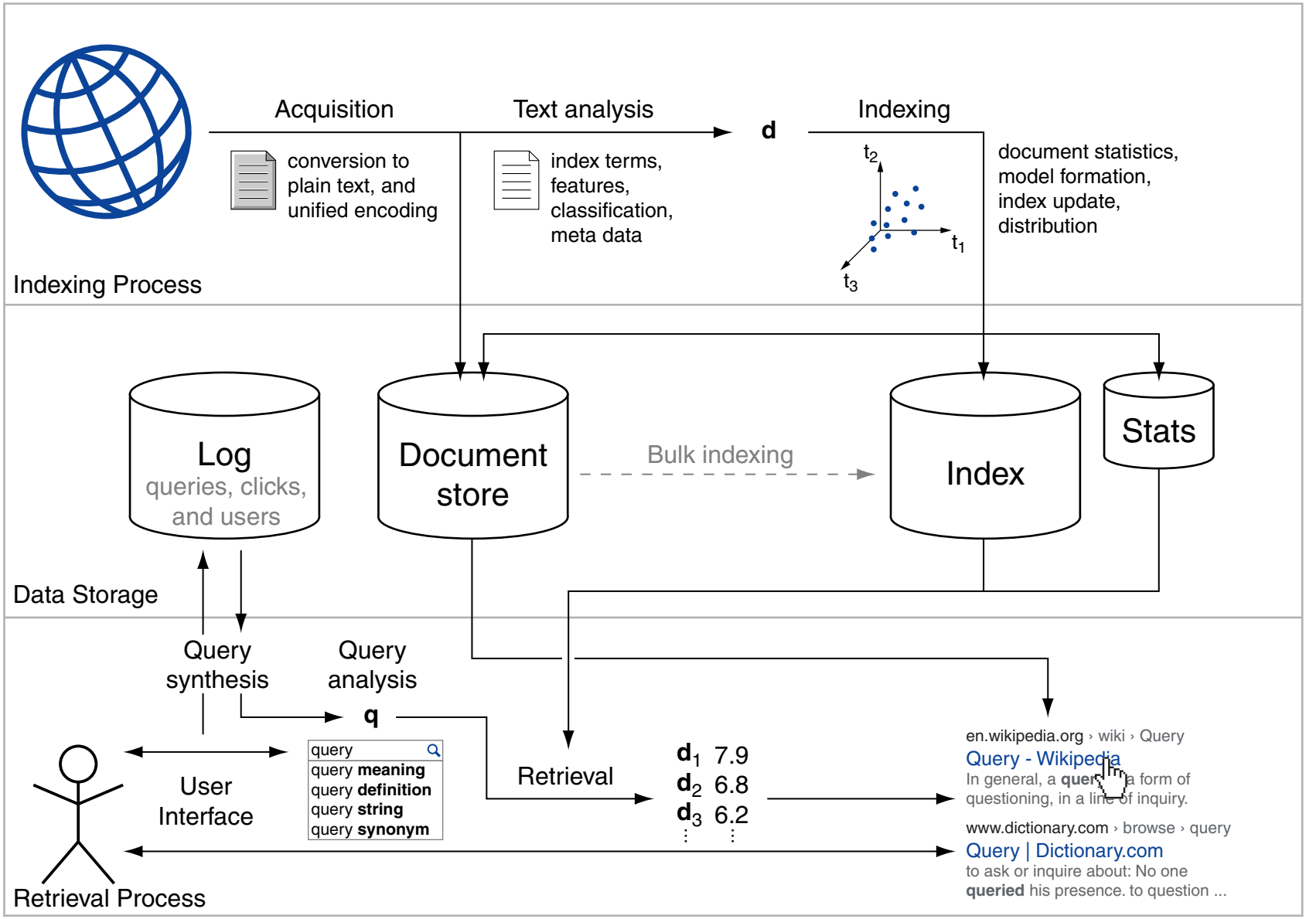
Relevance feedback refines a query in multiple retrieval runs:

1. Retrieval of documents using the current query.
2. Identification of relevant / irrelevant documents among the top-ranked ones.
3. Extraction of terms related to the user's information need from the relevant documents.
4. Addition the extracted terms to the query; next retrieval run.

Sources of relevance feedback:

- Direct / explicit relevance feedback
  A user marks retrieved documents as relevant or irrelevant.
- 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.





Retrieval

Given a query representation, the ranking step scores and orders the documents indexed 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 and queries, and a function to rank them based on the representations.

Document representations are typically pre-computed offline and indexed (e.g., the term weights under $tf \cdot idf$). Rankings are computed online for each query.

Key components:

- Document Scoring
- Distribution

Retrieval

Document Scoring

The scoring step quantifies the relevance of the documents indexed to a query.

Retrieval

Document Scoring

The scoring step quantifies the relevance of the documents indexed to a query.

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 document scoring function ρ is:

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

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

Retrieval

Document Scoring

The scoring step quantifies the relevance of the documents indexed to a query.

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 document scoring function ρ is:

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

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

Observations:

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

Retrieval

Document Scoring

Computing document scores requires index access. Different access strategies govern what can be accomplished and how the data has to be organized. The two most salient strategies are as follows:

- Document-at-a-time scoring
  - Precondition: a total order of documents in the index's postings lists is enforced (e.g., ordering criterion: document ID or rather document quality).
  - Postlists of a query's terms are traversed in parallel to score one document at a time.
  - 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
  - Traverse postlists one a time (e.g., term ordering criterion: frequency or importance).
  - Maintain temporary query postlist, containing candidate documents.
  - As each document's score accumulates, 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).

Retrieval

Distribution

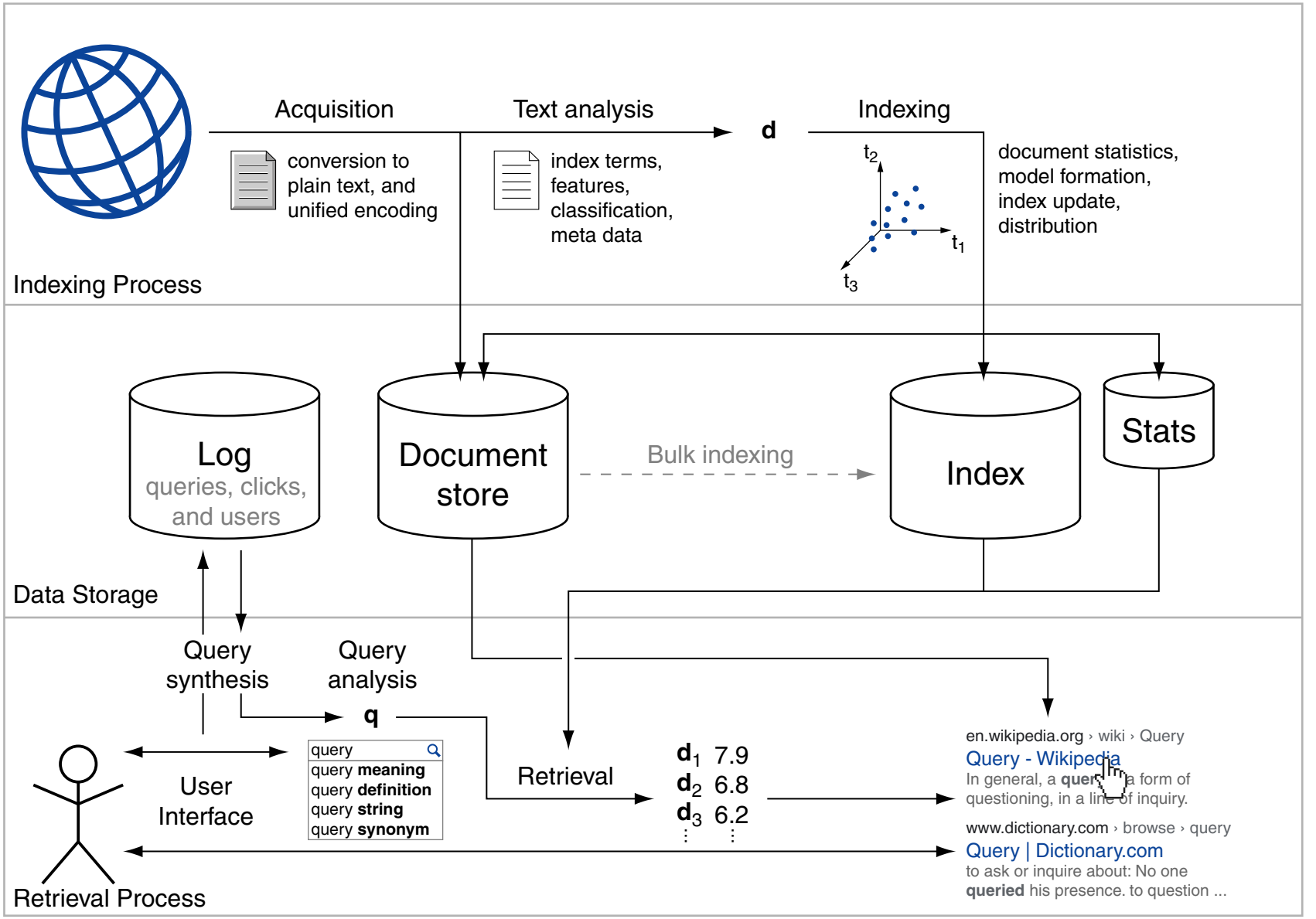
The distribution of query processing depends on that of the index.

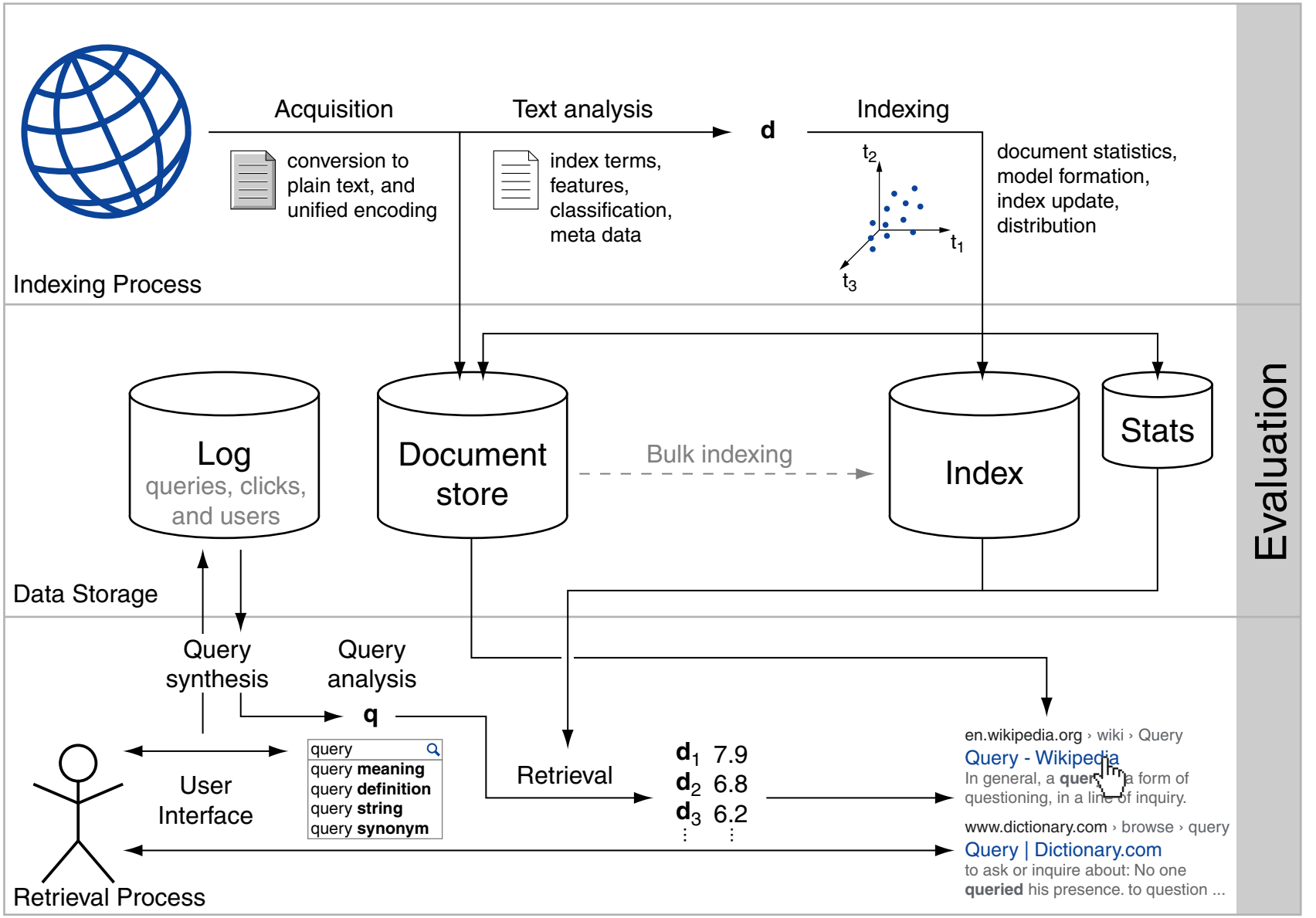
- Query broker / load balancer
  Decides which shard and which replicated copies to access. Receives and merges results.
- Cache
  Keeps frequently used data close at hand (e.g., in main memory) to reduce latency. Caches may include indexes containing important documents only held in main memory, precomputed search results, temporary postlists, parts of postlists, and the optimized usage of caching hierarchy from operating system to hardware caches.

Chapter IR:II

II. Architecture of a Search Engine

- Overview
- Acquisition
- Text Analysis
- Indexing
- User Interface
- Query Analysis and Synthesis
- Retrieval
- Evaluation





Evaluation

Overview

Evaluation addresses the analysis of search effectiveness and efficiency.

- Retrieval analysis
  - Definition of goals (e.g., relevant documents first, diversity, novelty, etc.)
  - Acquisition of relevance judgments for query-document pairs (e.g., via crowdsourcing)
  - Measurement theory (e.g., strong emphasis on top results is common for web search)
  - Log analysis of recorded search behavior
  - User studies and A/B tests
- User experience analysis
  - Definition of goals (e.g., usability, user satisfaction, etc.)
  - Log analysis of recorded user behavior
  - User studies and A/B testing
- Runtime analysis
  - Definition of goals (e.g., throughput, response time, etc.)
  - Log analysis of recorded system behavior
  - Lab experiments and simulation

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]

Architecture of a Search Engine