**Indices and Boolean retrieval**

**Precision**

* Fraction of the retrieved docs that are relevant to the user’s information need.

**Recall**

* Fraction of the docs in collection that are retrieved.

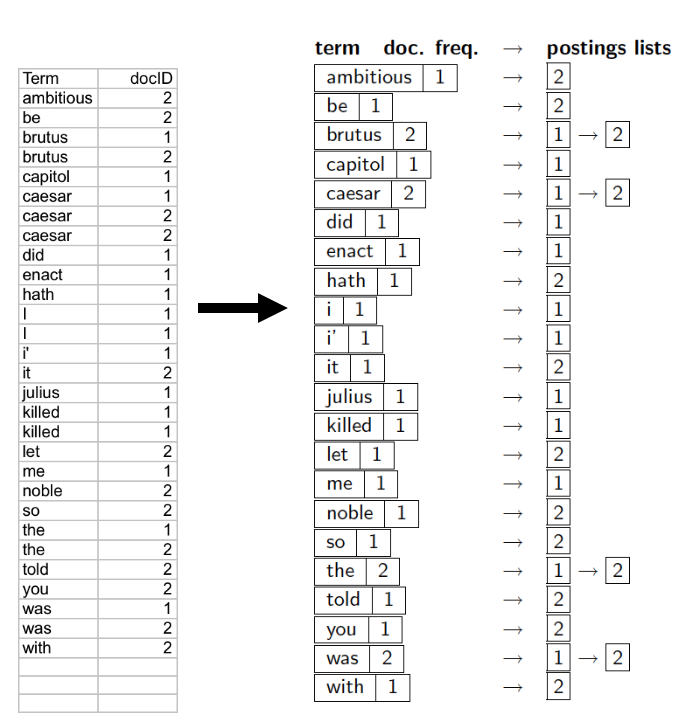
**Term-document incidence matrices:**

* Can generate incidence vectors for each term.
* Answer queries by bitwise AND/OR operation.
* Impractical in real-life due to sparseness.

**Inverted index**

* For each term t, we must store a list of all documents that contain t. (Posting list)
  + This list is sorted by docID.
* Identify each document by docID, a document serial number.

**Inverted index construction**

* **Initial stages**
  + Tokenization
    - Cut character sequence in to word tokens.
  + Normalization
    - Map text and query term to the same form. (**U.S.A** and **USA**)
  + Stemming
    - We may wish different forms of a word root to match. (**Authorize, authorization**)
  + Stop words
    - We may omit very common words. (**the, or, a, of**)
* **Token sequence processing**
  + Sort by terms and then by docID.
* **Construct dictionary and postings**
  + Multiple term entries in a single document are merged.
  + Split into dictionary and postings.
  + Document frequency information is added. (Number of different documents that contains a term)

**Query processing**

* Retrieve the posting lists of the query terms.
* Merge the posting lists together. (AND/OR)
  + AND: Construct a new posting list that only contains the postings that are in both.
  + OR: Construct a new posting list that contains all the postings.

**Boolean retrieval model**

* The **Boolean retrieval model** is being able to ask a query that is a Boolean expression
  + Boolean Queries are queries using AND, OR and NOT to join query terms.
    - Views each document as a set of words.
    - Is precise: document matches condition or not.
  + Perhaps the simplest model to build an information retrieval system on.
* Primary commercial retrieval tool for 3 decades.
* Many search systems are still using the Boolean retrieval model:
  + Email, library catalog, etc.

**Query evaluation optimizations**

* Process query terms in order of increasing frequency/length
  + Eliminate most of the postings during the earlier stages of the process.
* General optimization
  + (A OR B) AND (C OR D)
  + Get document frequency for all terms
  + Estimate the size of each OR clause by the sum of its document frequencies.
  + Process in increasing order of OR clause sizes.

**Phrase queries**

* We want to be able to answer queries such as ‘’Stanford university’’ as a phrase.
* Thus, the sentence ‘’I went to university at Stanford’’ is not a match.
  + The concept of phrase queries has proven easily understood by users; one of the few ‘’advanced search’’ ideas that works.
  + Many other queries are implicit phrase queries.
* For this. It is no longer suffices to store only <term: docs> entries.

**Solution 1: Biword indexes**

* + Index every consecutive pair of terms in the text as a phrase.
  + For example, the text ‘’Friends, Romans, Countrymen’’ would generate the Biwords:
    - Friends romans
    - Romans, countrymen
  + Each of these Biwords is not a dictionary term
  + Two-word phrase query processing is now immediate
  + **Longer phrase queries:**
    - Longer phrases can be processed by breaking them down
    - ‘’Stanford university palo alto’’ can be broken into the Boolean query on Biwords
      * **Stanford university** AND **university palo** AND **palo alto**

**Issues with Biword indexes**

* + False positives
    - Without the document contents, we cannot verify that the documents matching the Boolean query do contain the phrase. The document could contain all the Biwords, but not as a coherent phrase.
  + Index blowup due to bigger dictionary
    - Infeasible for more than Biwords, big even just for Biwords
  + Biwords indexes are not the standard solution but can be part of a compound query evaluation strategy.

**Solution 2: Positional indexes**

* + In the postings, store for each term, the positions in which the tokens of i appear.
    - Term: frequency,
      * Doc1: pos1, pos2...
      * Doc2: pos1, pos2...
  + A position is the order of the token, not the index in the content of the document. Position 15 means the token is the 15th token in the normalized document content.
  + **Processing a phrase query**
    - Phrase **‘’to be or not to be’’**
    - Terms: to, be, or, not
    - Extract inverted index entries for each distinct term
    - Merge their doc:position lists to enumerate all positions with ‘**’to be or not to be**’’.
      * To: 2: 1,17...., 4: 8, **16**, 190.......
      * Be: 1: 17...., 4: **17**, 197......
    - Same general method for proximity searches.
  + **Proximity queries**
    - **LIMIT /3 STATUTE /3 FEDEAL /2 TORT**
      * /k means ‘’within k words of’’.
    - Clearly, positional indexes can be used for such queries, Biword indexes cannot.
  + **Positional index size**
    - A positional index expands postings storage substantially
      * Even though indices can be compressed
    - Nevertheless, a positional index is not standardly used because of the power and usefulness of phrase and proximity queries, whether used explicitly or implicitly in a ranking retrieval system.
    - Need an entry for each occurrence, not just once per document.
    - Index size depends on average document size
      * Average web page has < 1000 terms
      * SEC filings, book, even some epic poems, easily 100,000 terms.
    - **Rule of thumb**
      * Positional index is 2-4 time as large as a non-positional index
      * Positional index size is 35%-50% of volume of original text.
      * **(For all ‘’English like’’ languages)**

**Combination schemes**

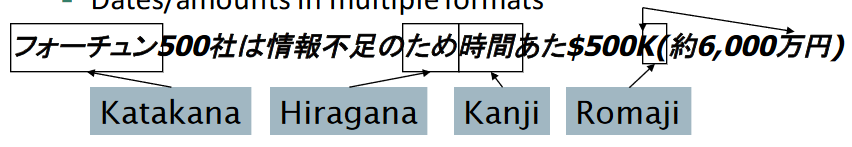
* + Biword indexes and positional indexes can be profitably combined
    - For particular phrases, such as named entities (‘’Michael Jackson’’, ‘’Britney Spears’’) it is inefficient to keep on merging positional posting lists.
      * Even more so for phrases like ‘’The Who’’(band)

**Document ingestion**

**Parsing a document**

* Format (pdf/word/excel/html……)
* Language
* Character set (CP1252, UTF-8, Unicode……)
* **Complications**
  + Documents being indexed can include documents from many different languages
    - A single index may contain terms from many languages.
  + Sometimes a document or its components can contain multiple languages/formats
    - French email with German pdf attachment
    - French email quote clauses from an English-language contract
  + What is a unit document?
    - A file?
    - An email? (Perhaps one of many in a single mbox file)
      * What about an email with 5 attachments?
    - A group of files (e.g., PPT or LaTeX split over HTML pages)

**Tokens – Tokenization**

* A token is an instance of a sequence of characters.
* Each such token is now a candidate for an index entry, after further processing.
* **Issues in tokenization:**
  + “Finland’s capital” ->
    - Finland And s? Finlands? Finland’s?
  + “Hewlett-Packard” ->
    - Hewlett and Packard as tow tokens?
  + San Francisco
    - One token or two?
  + Numbers
    - Often have embedded spaces
    - Older information retrieval systems may not index numbers
      * However often very useful, things like looking up error codes/stack traces on the web
      * Often index “meta-data” separately (creation data, format, etc.)
  + Language issues
    - French
      * L’ensemble
        + One token or two tokens?
    - German
      * “Lebenscersicherungsgesellschaftsangestellter”
      * ‘Life insurance company employee’
      * German retrieval systems benefit greatly from a **compound splitter** module
    - Chine and Japanese have no spaces between words
      * Not always guaranteed a unique tokenization
    - Further complicated in Japanese, with multiple alphabets intermingled.
      * Dates/amounts in multiple formats
    - Arabic (or Hebrew) is basically written right to left, but with certain items like numbers written left to right.
      * Words are separated, but letter forms within a word form complex ligatures.
      * With Unicode, the surface presentation is complex, but the stored form is straight forward.

**Terms – The things indexed in an IR system**

* **Stop words**
  + Words that have little sematic content: the, a, and, to, be
  + There are a lot of them.
  + Can be excluded from the dictionary entirely.
  + However, the trend is away from excluding them
    - Good compression techniques mean the space for including stop words in a system is very small
    - Good query optimization techniques mean you pay little ay query time for including stop words
    - You need them for
      * Phrase queries: “King of Denmark”
      * Various song title, etc.:” Let it be”, “To be or not to be”
      * “Relational queries”: “flights to London”
* **Normalization to terms**
  + We may need to “normalize” words in indexed text as well as query words into the same form.
    - We want to match U.S.A with USA
  + Result is terms: a **term** is a (normalized) word type, which is an entry in our IR system dictionary.
  + We most commonly implicitly define equivalence classes of terms by
    - Deleting periods to form a term
      * U.S.A, USA
    - Deleting hyphens to form a term
      * Anti-discriminatory, antidiscriminatory
  + Other languages
    - Accents: French Résumé vs resume.
    - Umlauts: German: Tuebingen vs Tűbingen
    - Even languages that standardly have accents, users often may not type them
  + **Tokenization and normalization may depend on the language and so is intertwined with language detection**
  + **Crucial: need to “normalize” indexed text as well as query terms identically.**
  + An alternative to equivalence classing is to do asymmetric expansion
    - An example of where this may be useful:
      * Enter: window
        + Search: window, windows
      * Enter: windows
        + Search: Windows, windows, window
      * Enter: Windows
        + Search: Windows
    - Potentially more powerful, but less efficient
* **Case folding**
  + Reduce all letters to lower case
    - Exception: upper case in mid-sentence?
      * General Motors
      * Fed vs fed
      * SAIL vs sail
    - Often best to lower case everything, since users will use lowercase regardless of “correct” capitalization
* **Thesauri and Soundex**
  + Do we handle synonyms and homonyms?
    - Hand-constructed equivalence classes
      * Car = automobile color = colour
    - We can rewrite to form equivalence class terms
      * When the document contains automobile, index is under car-automobile (and vice-versa)
    - We can expand a query
      * When the query contains automobile, look under car and automobile
  + What about spelling mistakes?
    - One approach is Soundex, which forms equivalence classes of words based on phonetic heuristics.

**Stemming and lemmatization**

**Lemmatization**

* Reduce inflectional/variant forms to base form
* Examples:
  + Am, are, is -> be
  + Car, cars, car’s, cars’ -> car
  + The boy’s cars are different colors -> the boy car be different color
* Lemmatization implies doing “proper” reduction to dictionary headword form

**Stemming**

* Reducer terms to their “roots” before indexing
* “Stemming” suggests crude affix chopping
  + Language dependent
  + Examples:
    - Automat, automatic, automation -> automat
* **Porter’s algorithm**
  + Most common algorithm for stemming English
  + Conventions + 5 phases of reduction
    - Phases applied sequentially
    - Each phase consists of a set of commands
    - Sample convention
      * Of the rules in a compound command, select the one that applies to the longest suffix
    - Typical rules in Porter
      * Sses -> ss
      * Ies -> i
      * Ational -> ate
      * Tional -> tion
    - Weight of word sensitive rules
      * (m > 1) EMENT -> ø (EMENT goes to the empty string, if more than one character is left after the removal)
        + Replacement -> replac
        + Cement -> cement (unchanged)
* Other stemmer
  + Lovins stemmer
    - Single-pass, lonest suffix removal (250 rules)
  + Paice/Husk stemmer
  + Snowball
  + Full morphological analysis (lemmatization), At most modest benefits for retrieval.
* Language and application specific
* Does stemming help?
  + English: Very mixed results. Helps recall but harms precision.
  + Useful for Spanish, German Finnish….