CHOOSING A DOCUMENT UNIT

- Determine what the document unit for indexing is.
- For very long documents, the issue of indexing granularity arises.
- For example: for a collection of books, it would usually be a bad idea to index an entire book as a document.
- A search for "Chinese toys" might bring up a book that mentions "China" in the first chapter and "toys" in the last chapter but this does not make it relevant to the query.
- Instead we may well wish to index each chapter or paragraph as mini-document.
- Matches are then more likely to be relevant.

TOKENIZATION

- Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces called tokens.
- So it is the process of breaking a stream of text up into words, phrase, symbols or other meaningful element called tokens.
- The list of tokens becomes input for further processing such as parsing, text mining, etc.
- During this phase all remaining text is parsed, lowercased and all punctuation removed.
- For example: Input: Friends, Romans, Countrymen, Lend me your ears

  Output: |Friends| |Romans| |Countrymen| |Lend| |me| |your| |ears|
- A token is an instance of characters in some particular document that are grouped together as a useful semantic unit for processing.
- A type is a class of all tokens containing the same character sequence.
- A term is a type that is included in the IR systems dictionary, i.e. a term means a normalized document.
- For example: if document to be indexed is 'to sleep per chance to dream'
  - There are 5 tokens.
  - There are 4 types (2 instances of "to").
  - There are 3 terms ("to" is defined as stop word)

- Issues of tokenization are language specific.
- It thus requires the language of the document to be known.
- Computer technology has introduced new types of character sequences that a tokenizer should probably tokenize as a single token.
- For example: email id → gblack@gmail.com
  URLs → http://stuf.big.com/new/special .htm
  IP address → 192.168.0.1
- In English, hyphenation is used for various purposes ranging from splitting up vowels in words (co-education), to joining noun as names (Hewlett-Packaged).
- The first one can be regarded as one token (co-education), but difficult in second one.

DIFFICULTIES OF TOKENIZATION

- Splitting on white spaces can also split what should be regarded as a single token.
- Splitting on spaces can cause bad retrieval result.
- Example: search for "York University" mainly returns documents containing "New York University".
- Regarding to hyphen and space, a query for "over-eager", should search for "over-eager" OR "over eager" OR "overeager".
- Each new language presents some new issues.
- The languages like Chinese, Japanese; there is no space as splitter.
- In such cases, we use word segmentation.
- Segmentation is the method of taking the longest vocabulary match with some heuristic for unknown words to use of machine learning such as HMM (Hidden Marker Model).

DROPPING COMMON TERMS (STOP WORDS)

- Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary.
- These words are called stop words.

- The general strategy for determining a stop list is to sort the terms by collection frequency and then to take the most frequent terms and are then discarded during indexing.
- Using a stop list significantly reduce the number of postings that a system has to store.
- For example: a, am, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with, etc.
- A lot of time not indexing stop words does little harm.
- For example: the phrase query "President of the United States" which contains two stop words, is more precise that "President" AND "United States".
- Also, the meaning of "flights to London" is likely to be lost if the word "to" is stopped out.
- A search for Vannevar Bush's article "As we may think" will be difficult if the first three stopped words are dropped and the system simple search for documents containing word "think".
- Some song titles (to be or not to be, let it be, I don't want to be) are common with stop words.
- So IR system has focused precisely on how we can exploit the statistics of language so as to be able to cope with common words in better ways.

NORMALIZATION (EQUIVALENCE CLASSING OF TERMS)

- After document tokenization, we have to match query tokens to documents token lists, but it is somehow difficult.
- There are cases, where tokens are not quite the same, but we still want to match them.
- Example: U.S.A should match (USA) or even (US).
- Token normalization is about transforming tokens into a standard form.
- This allows matches to occur despite superficial differences.
- Usual way to normalize is to create equivalence classes.
- Example: anti-discriminating and anti-discriminatory are both in same class, so that searches for one term will retrieve documents that contain either.
- Alternative to equivalence classes are explicit rules.
- Example: window → window, windows
  windows → Windows, windows

- But some normalization may do more harm than good.
- Example: WHO → who

STEMMING AND LEMMATIZATION

- For grammatical reasons, documents are going to use different forms of a word such as organize, organizes and organizing.
- Additionally, there are families of derivationally related words with similar meanings such as democracy, democratic and democratization.
- The goal of both stemming and lemmatization is to reduce inflectional forms.
- Example: am, are, is → be

  car, cars, car's, cars' → car
- Example: the result of this mapping of text will be something like "the boy's cars are different colors" → "the boy car be differ color".
- Stemming is defined as crude heuristic process that chops off the ends of words.
- Language dependent.
- Works quite well for English language.
- Example: Automate automatic, automation all reduce to automat.
- Lemmatization usually refers to doing things properly with the use of vocabulary and morphological analysis of words.
- For example: with the token "saw", stemming might return "s" whereas lemmatization would attempt to return "see".

PORTER ALGORITHM

- Most common algorithm for stemming English.
- Result suggests that is at least as good as other stemming option.
- Removing suffixes by automatic means is an operation which is especially useful in the field of IR.
- Terms with a common stem will usually have similar meanings.
- For example: CONNECT, CONNECTED, CONNECTING, CONNECTION, CONNECTIONS.

- The performance of an IR system will be improved if term groups are conflated into a single term.
- This may be done by removal of the various suffixes "ED", "ING", "ION", "IONS" to leave the single term CONNECT.
- A consonant in a word is a letter other than A, E, I, O or U and other than Y preceded by a consonant.
- Example: in "TOY", the consonant are T and Y.
- In "SYZYGY", they are S, Z and G.
- If a letter is not a consonant, it is a vowel.
- A consonant is denoted by c, a vowel is denoted by v.
- A list ccc… of length greater than 0 will be denoted by c.
- A list vvv… of length greater than 0 will be denoted by v.
- Any word has one of the following forms: c…c, c…v, v…v, v…c
- These may all be represented by the single form: [c]vcvc…[v], where the square brackets denote arbitrary presence of their contents.
- This may again be written as [c] $(vc)^m$ [v], where, m is called the measure of word or word part represented in "vc" form.
- Examples: m = 0 TR, EE, TREE, Y, BY

  m = 1 TROUBLE, OATS, TREES, IVY

  m = 2 TROUBLES, PRIVATE, OATEN, ORRERY
- The rules for removing a suffix will be given in the form:

  <condition> $S_1 \rightarrow S_2$
- Example:

Rules	Example
SSES → SS	caresses → caress
IES → I	ponies → poni
SS → SS	caress → caress
S →	cats → cat
(m > 1) EMENT	Replacement → replace (btu not cement → c) cement → cement

PHRASE QUERIES

- We want to answer a query such as “Stanford University” as a phrase.
- Thus, “the inventor Stanford Orshinsky never went to university” shouldn’t be matched.
- About 10% of web queries are phrase queries.

BUILDING AN INVERTED INDEX

- Inverted index, also called postings file or inverted file, is an index data structure storing a mapping from content, such as words or numbers to its locations in a database file or in a document or a set of documents.
- The purpose of an inverted index is to allow fast full text searches.
- An index always maps back from terms to the parts of a document where they occur.
- A dictionary of terms is kept.
- Then for each term, a list is maintained in which documents the term occurs in.
- Each item in the list which records that a term appeared in a document is called a posting.
- The list is then posting list.
- The dictionary will be sorted alphabetically and each postings list is sorted by document ID.
- Example: DOC 1 = new home sales top forecasts

  DOC 2 = home sales rise in July

  DOC 3 = increase in home sales in July

  DOC 4 = July new home sales rise

  forecasts → |DOC 1|

  home → |DOC 1|→|DOC 2|→|DOC 3|→|DOC 4|→ posting list

  increase → |DOC 3|

  July → |DOC 2|→|DOC 3|→|DOC 4|→ increasing

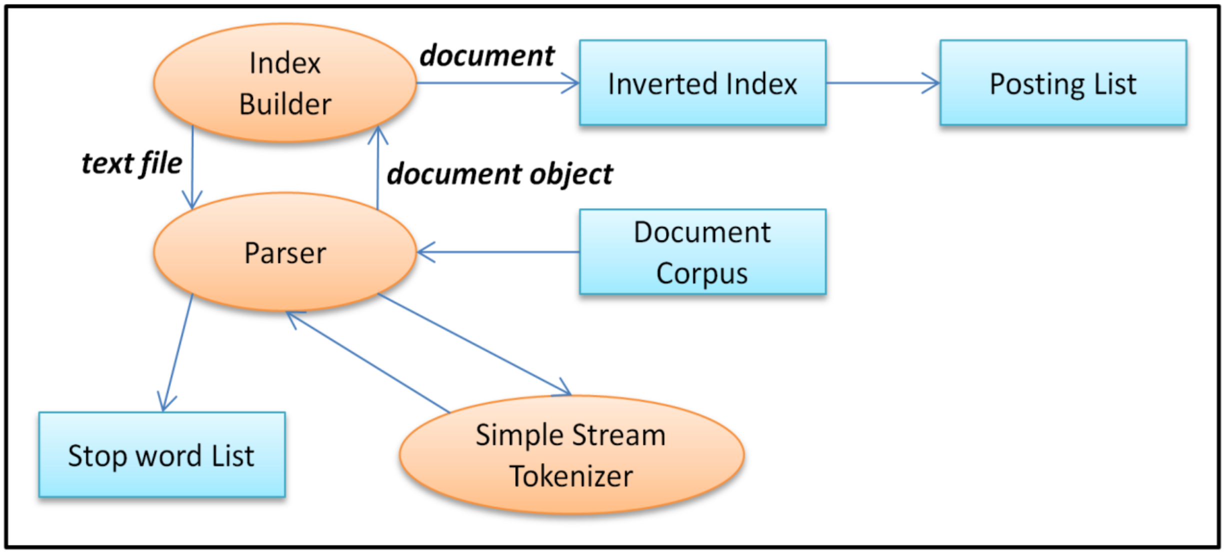
  new → |DOC 1|→|DOC 4|

  rise → |DOC 2|→|DOC 4|

  sales → |DOC 1|→|DOC 2|→|DOC 3|→|DOC 4|

  top → |DOC 1|

INDEXING ARCHITECTURE



BIWORD INDEX

- Index every consecutive pair of terms in the text as a phrase.
- Example: Friends, Romans, Countrymen would generate two bi-words "Friends Romans" and "Romans Countrymen".
- Each of these bi-word is now a vocabulary term.

POSITIONAL INDEXES

- Posting lists in a positional index in which each posting is a docID and a list of positions.
- Example:

  Cat, 100

  <1, 6 :<7, 18, 33, 72, 86, 231>;

  2, 5 : <1, 17, 74, 222, 255>;

  4, 2 : <8, 16>;

  ..

  ..

  >

- The word “cat” has a document frequency 100 and occurs 6 times in document 1 at positions 7, 18, 33, 72, 86, 231 and so on.

SPARSE VECTORS

- Most documents and queries do not contain most word, so vectors are sparse.
  i.e. most entries are zero (0).
- Need efficient methods for storing and computing with sparse vectors.
- We can use sparse vectors as lists, sparse vectors as trees, sparse vectors as Hash Table.