The major steps in inverted index construction

* Collect the documents to be indexed
* Tokenize the text
* Do linguistic preprocessing of tokens – deals with building equivalence classes of tokens which are set of terms that are indexed
* Index the documents that each term occurs in

Recommendation found in the book: “An IR system should be design to offer choices of granularity” – they refer to the fact that we should choose a different unit for indexing when using large documents (for example index paragraphs, instead of the whole book) than when indexing small documents (as we have in our case I guess); for the moment going with the whole e-mail I think is ok, but I say that we need to keep this in mind as a possible solution for some problems that might appear in the future

Pre-processing steps

1. Tokenization

* The task of chopping it up into pieces (tokens), perhaps at the same time throwing away certain characters, such as punctuation
* Token = an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing
* Type = is the class of all tokens containing the same character sequence
* Term = is a type that is included in the IR’s dictionary

1. Stop-word removal – Lucene uses a default list of stop words that are removed - [link](https://lucene.apache.org/core/old_versioned_docs/versions/3_0_1/api/core/org/apache/lucene/analysis/standard/StandardAnalyzer.html%23version) ; it is possible to use a custom list of words but I suggest to keep it default so we don’t deal with natural language processing

* Words which would appear to be of little value in helping select documents matching a user need
* General strategy: sort the terms by collection frequency and then to take the most frequent terms as a stop list
* Reduces the number of postings that a system has to store – check Chapter 5 /87 from the book
* Can do harm in phrase searches, for example “flight to London”, song titles,\
* The general trend in IR systems over time has been from standard use of quite large stop lists to very small stop lists, to no stop list
* web search engines generally do not use stop lists
* Usually for coping with stop words more advanced techniques are used:
  + Good compression for storing the postings (that now are usually large)
  + Term weighting leads to common words to have little impact on document rankings
  + An IR system with impact-sorted indexes can terminate scanning a postings list early when weights get small – common words do not cause a large additional processing cost for the average query

1. Normalization

* The process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences of the tokens
* The standard way to normalize is to implicitly create equivalence classes, which are normally named after one member of the set
* You can maintain relations between un-normalized tokens and this can be done in 2 ways:
  + Index un-normalized tokens and maintain q query expansion list of multiple vocabulary entries to consider for a certain query term
  + Perform the expansion during index construction
  + Both methods have disadvantages
* Capitalization/case-folding
* Accents and diacritics

1. Stemming and lemmatization

* They want to reduce the inflectional forms and sometimes derivationally related forms of a word to a common base form: am, are, is => be; car, cars, car’s, cars => car
* Stemming refers to a crude heuristic process that chops off the ends of the words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivation affixes
* Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word
* The most common algorithm for stemming English is “Porte’s algorithm”
* Stemming increases recall while harming precision

1. Advanced analysis

A useful site that I found is [link](http://futupark.wordpress.com/2008/04/01/text-analysis-inside-lucene/).

The StandardAnalyzer can identify certain kinds of tokens, such as company names, email addresses and hostnames. It also lowercases each token and removes stop words and punctuation.

We can specify an analyzer per document. Each analyzed field of every document indexed with the IndexWriter instance uses the analyzer specified by default. But if an individual document has special analysis needs, the analyzer may be specified on a per-document basis: both the addDocument and updateDocument methods in IndexWriter optionally accept an analyzer to be used for that one document.

We can make our own analyzer, but in two weeks that’s mission impossible 5, I guess. If you decide to do so, maybe an analyzer which takes into consideration synonyms would make a good impression, but for this the hardest part is to define synonyms by hand I guess. Adding synonyms could be put in “advanced analysis” category. I found this [link](https://lucene.apache.org/core/old_versioned_docs/versions/3_0_1/api/contrib-wordnet/org/apache/lucene/wordnet/SynonymMap.html) which I think it uses Wordnet for getting synonyms. Maybe we can use it.

More information about StandardAnalyzer:

“StandardAnalyzer holds the honor as the most generally useful built-in analyzer. A JFlex-based1 grammar underlies it, tokenizing with cleverness for the following lexical types: alphanumerics, acronyms, company names, email addresses, computer hostnames, numbers, and words with an interior apostrophe, serial numbers, IP addresses, and Chinese and Japanese characters. StandardAnalyzer also includes stop-word removal, using the same mechanism as the StopAnalyzer (identical default English set, and an optional Set constructor to override). StandardAnalyzer makes a great first choice. Using StandardAnalyzer is no different than using any of the other analyzers, as you can see from its use in section 4.1.1 and AnalyzerDemo (listing 4.1). Its unique effect, though, is apparent in the different treatment of text. For example, compare the different analyzers on the phrase “XY&Z Corporation - xyz@example.com” from section 4.1. StandardAnalyzer is the only one that kept XY&Z together as well as the email address xyz@example.com; both of these showcase the vastly more sophisticated analysis process. “