Indexing

• An *index* is a list of things (keys) with pointers to other things (items).
  – Keywords → catalog numbers (→ shelves).
  – Concepts → page numbers.
  – Terms → documents.
• Need for indexes:
  – Ease of use.
  – Speed.
  – Scalability.
Manual vs. Automatic Indexing

• Manual:
  – An “expert” assigns keys to each item.
  – Example: card catalog.

• Automatic:
  – Keys automatically identified and assigned.
  – Example: Google.

• Automatic as good as manual for most purposes.

Text Processing

• First step in automatic indexing.
• Converting documents into *index terms*.
• Terms are not just words.
  – Not all words are of equal value in a search.
  – Sometimes not clear where words begin and end.
    • Especially when not space-separated, e.g. Chinese, Korean.
  – Matching the exact words typed by the user doesn’t work very well in terms of effectiveness.
Text Processing Steps

• For each document:
  – Parse it to locate the parts that are important.
  – Segment and tokenize the text in the important parts to get words.
  – Remove stop words.
  – Stem words to common roots.

• Advanced processing may include phrases, entity tagging, link-graph features, and more.

Parsing

• Some parts of a document are more important than others.

• Document parser recognizes structure using *markup* such as HTML tags.
  – Headers, anchor text, bolded text are likely to be important.
  – JavaScript, style information, navigation links less likely to be important.
  – Metadata can also be important.
Example Wikipedia Page

Wikipedia Markup

```
<title>Tropical fish</title>

Tropical fish include [[fish]] found in [[Tropics|topical]] environments around the world, including both [[fresh water|freshwater]] and [[sea water|salt water]] species. [[Fishkeeping|Fishkeepers]] often use the term "tropical fish" to refer only those requiring fresh water, with saltwater tropical fish referred to as "[[list of marine aquarium fish species|marine fish]]".
```
Document Parsing

- HTML pages organize into trees.

Nodes contain blocks of text.

- <HTML>
  - <HEAD>
    - <TITLE>
      - Tropical fish
    - <META>
    - <BODY>
      - <H1>
        - Tropical fish
      - <B>
        - Tropical fish
      - <A>
        - Tropical fish
      - <P>
        - Fish found in environments around the world
      - <A>
        - Tropical fish
End Result of Parsing

• Blocks of text from important parts of page.
  – Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species. Fishkeepers often use the term “tropical fish” to refer only those requiring fresh water, with saltwater tropical fish referred to as “marine fish”.

• Next step: segmenting and tokenizing.

Tokenizing

• Forming words from sequence of characters in blocks of text.
• Surprisingly complex in English, can be harder in other languages.
• Early IR systems:
  – Any sequence of alphanumeric characters of length 3 or more.
  – Terminated by a space or other special character.
  – Upper-case changed to lower-case.
Tokenizing

• Example:
  – “Bigcorp's 2007 bi-annual report showed profits rose 10%.” becomes
  – “bigcorp 2007 annual report showed profits rose”

• Too simple for search applications or even large-scale experiments

• Why? Too much information lost
  – Small decisions in tokenizing can have major impact on effectiveness of some queries

Tokenizing Problems

• Small words can be important in some queries, usually in combinations
  • xp, ma, pm, ben e king, el paso, master p, gm, j lo, world war II

• Both hyphenated and non-hyphenated forms of many words are common
  – Sometimes hyphen is not needed
    • e-bay, wal-mart, active-x, cd-rom, t-shirts
  – At other times, hyphens should be considered either as part of the word or a word separator
    • winston-salem, mazda rx-7, e-cards, pre-diabetes, t-mobile, spanish-speaking
Tokenizing Problems

- Special characters are an important part of tags, URLs, code in documents
- Capitalized words can have different meaning from lower case words
  - Bush, Apple
- Apostrophes can be a part of a word, a part of a possessive, or just a mistake
  - rosie o'donnell, can't, don't, 80's, 1890's, men's straw hats, master's degree, england's ten largest cities, shriner's

Tokenizing Problems

- Numbers can be important, including decimals
  - nokia 3250, top 10 courses, united 93, quicktime 6.5 pro, 92.3 the beat, 288358
- Periods can occur in numbers, abbreviations, URLs, ends of sentences, and other situations
  - I.B.M., Ph.D., cis.udel.edu
- Note: tokenizing steps for queries must be identical to steps for documents
Tokenizing Process

• Assume we have used the parser to find blocks of important text.
• A word may be any sequence of alphanumeric characters terminated by a space or special character.
  – everything converted to lower case.
  – everything indexed.
• Defer complex decisions to other components
  – example: 92.3 → 92 3 but search finds documents with 92 and 3 adjacent
  – incorporate some rules to reduce dependence on query transformation components

End Result of Tokenization

• List of words in blocks of text.
  – tropical fish include fish found in tropical environments around the world including both freshwater and salt water species fishkeepers often use the term tropical fish to refer only those requiring fresh water with saltwater tropical fish referred to as marine fish
• Next step: stopping.
• But first: text statistics.
Text Statistics

• Huge variety of words used in text but
• Many statistical characteristics of word occurrences are predictable
  – e.g., distribution of word counts
• Retrieval models and ranking algorithms depend heavily on statistical properties of words
  – e.g., important words occur often in documents but are not high frequency in collection

Zipf’s Law

• Distribution of word frequencies is very skewed
  – a few words occur very often, many words hardly ever occur
  – e.g., two most common words (“the”, “of”) make up about 10% of all word occurrences in text documents
• Zipf’s “law”:
  – observation that rank (r) of a word times its frequency (f) is approximately a constant (k)
    • assuming words are ranked in order of decreasing frequency
  – i.e., \( r.f \approx k \) or \( r.P_r \approx c \), where \( P_r \) is probability of word occurrence and \( c \approx 0.1 \) for English
Zipf’s Law

Wikipedia Statistics (wiki000 subset)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total documents</td>
<td>5,001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total word occurrences</td>
<td>22,545,922</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vocabulary size</td>
<td>348,436</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Words occurring &gt; 1000 times</td>
<td>2,751</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Words occurring once</td>
<td>163,404</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word</th>
<th>Freq</th>
<th>r</th>
<th>Pr (%)</th>
<th>r.Pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>politician</td>
<td>5096</td>
<td>510</td>
<td>0.023</td>
<td>0.116</td>
</tr>
<tr>
<td>contractor</td>
<td>100</td>
<td>14,852</td>
<td>4.4 \times 10^{-4}</td>
<td>0.066</td>
</tr>
<tr>
<td>kickboxer</td>
<td>10</td>
<td>56,125</td>
<td>4.4 \times 10^{-5}</td>
<td>0.025</td>
</tr>
<tr>
<td>comedian</td>
<td>1</td>
<td>185,035</td>
<td>4.4 \times 10^{-6}</td>
<td>0.008</td>
</tr>
</tbody>
</table>
Top 50 Words from wiki000 Subset

<table>
<thead>
<tr>
<th>Word</th>
<th>Freq.</th>
<th>r</th>
<th>$P_r(%)$</th>
<th>$r.P_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>142490</td>
<td>1</td>
<td>0.063</td>
<td>0.063</td>
</tr>
<tr>
<td>of</td>
<td>832458</td>
<td>2</td>
<td>0.036</td>
<td>0.073</td>
</tr>
<tr>
<td>and</td>
<td>579322</td>
<td>3</td>
<td>0.025</td>
<td>0.077</td>
</tr>
<tr>
<td>in</td>
<td>505550</td>
<td>4</td>
<td>0.022</td>
<td>0.089</td>
</tr>
<tr>
<td>to</td>
<td>376854</td>
<td>5</td>
<td>0.016</td>
<td>0.083</td>
</tr>
<tr>
<td>a</td>
<td>357109</td>
<td>6</td>
<td>0.015</td>
<td>0.095</td>
</tr>
<tr>
<td>ref</td>
<td>264102</td>
<td>7</td>
<td>0.012</td>
<td>0.087</td>
</tr>
<tr>
<td>is</td>
<td>211077</td>
<td>8</td>
<td>0.009</td>
<td>0.074</td>
</tr>
<tr>
<td>s</td>
<td>183667</td>
<td>9</td>
<td>0.008</td>
<td>0.073</td>
</tr>
<tr>
<td>us</td>
<td>175823</td>
<td>10</td>
<td>0.007</td>
<td>0.075</td>
</tr>
<tr>
<td>for</td>
<td>144600</td>
<td>11</td>
<td>0.006</td>
<td>0.070</td>
</tr>
<tr>
<td>by</td>
<td>142221</td>
<td>12</td>
<td>0.006</td>
<td>0.075</td>
</tr>
<tr>
<td>was</td>
<td>119516</td>
<td>13</td>
<td>0.005</td>
<td>0.068</td>
</tr>
<tr>
<td>on</td>
<td>113533</td>
<td>14</td>
<td>0.004</td>
<td>0.070</td>
</tr>
<tr>
<td>with</td>
<td>112296</td>
<td>15</td>
<td>0.004</td>
<td>0.074</td>
</tr>
<tr>
<td>that</td>
<td>111514</td>
<td>16</td>
<td>0.004</td>
<td>0.079</td>
</tr>
<tr>
<td>are</td>
<td>103415</td>
<td>17</td>
<td>0.004</td>
<td>0.078</td>
</tr>
<tr>
<td>from</td>
<td>95811</td>
<td>18</td>
<td>0.004</td>
<td>0.076</td>
</tr>
<tr>
<td>or</td>
<td>87167</td>
<td>19</td>
<td>0.003</td>
<td>0.073</td>
</tr>
<tr>
<td>it</td>
<td>73110</td>
<td>20</td>
<td>0.003</td>
<td>0.064</td>
</tr>
<tr>
<td>an</td>
<td>67643</td>
<td>21</td>
<td>0.003</td>
<td>0.063</td>
</tr>
<tr>
<td>at</td>
<td>66023</td>
<td>22</td>
<td>0.002</td>
<td>0.065</td>
</tr>
<tr>
<td>which</td>
<td>66570</td>
<td>23</td>
<td>0.002</td>
<td>0.067</td>
</tr>
<tr>
<td>name</td>
<td>65350</td>
<td>24</td>
<td>0.002</td>
<td>0.069</td>
</tr>
<tr>
<td>be</td>
<td>64169</td>
<td>25</td>
<td>0.002</td>
<td>0.071</td>
</tr>
</tbody>
</table>

Zipf’s Law for wiki000 Subset
Zipf’s Law

• What is the proportion of words with a given frequency?
  – Word that occurs $n$ times has rank $r_n = k/n$
  – Number of words with frequency $n$ is
    • $r_n - r_{n+1} = k/n - k/(n+1) = k/n(n+1)$
  – Proportion found by dividing by total number of words = highest rank $= k$
  – So, proportion with frequency $n$ is $1/n(n+1)$

---

### Zipf’s Law

• Example word frequency ranking

<table>
<thead>
<tr>
<th>Rank</th>
<th>Word</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>4999</td>
<td>objective</td>
<td>494</td>
</tr>
<tr>
<td>5000</td>
<td>albany</td>
<td>494</td>
</tr>
<tr>
<td>5001</td>
<td>defend</td>
<td>494</td>
</tr>
<tr>
<td>5002</td>
<td>appeals</td>
<td>493</td>
</tr>
<tr>
<td>5003</td>
<td>125</td>
<td>493</td>
</tr>
<tr>
<td>5004</td>
<td>lasting</td>
<td>493</td>
</tr>
<tr>
<td>5005</td>
<td>png</td>
<td>493</td>
</tr>
</tbody>
</table>

• To compute number of words with frequency 493
  – rank of “png” minus the rank of “defend”
  – $5005 - 5001 = 4$
Example

<table>
<thead>
<tr>
<th>Num. occurrences (n)</th>
<th>Predicted proportion (1/n(n+1))</th>
<th>Actual proportion</th>
<th>Actual number of words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.500</td>
<td>.469</td>
<td>163,404</td>
</tr>
<tr>
<td>2</td>
<td>.167</td>
<td>.151</td>
<td>52,672</td>
</tr>
<tr>
<td>3</td>
<td>.083</td>
<td>.070</td>
<td>24,272</td>
</tr>
<tr>
<td>4</td>
<td>.050</td>
<td>.045</td>
<td>15,685</td>
</tr>
<tr>
<td>5</td>
<td>.033</td>
<td>.030</td>
<td>10,437</td>
</tr>
<tr>
<td>6</td>
<td>.024</td>
<td>.022</td>
<td>7,832</td>
</tr>
<tr>
<td>7</td>
<td>.018</td>
<td>.017</td>
<td>5,962</td>
</tr>
<tr>
<td>8</td>
<td>.014</td>
<td>.014</td>
<td>4,890</td>
</tr>
<tr>
<td>9</td>
<td>.011</td>
<td>.011</td>
<td>3,886</td>
</tr>
<tr>
<td>10</td>
<td>.009</td>
<td>.009</td>
<td>3,291</td>
</tr>
</tbody>
</table>

- Proportions of words occurring \(n\) times in 5,001 Wikipedia documents
- Vocabulary size is 348,436.

Vocabulary Growth

- As corpus grows, so does vocabulary size
  - Fewer new words when corpus is already large
- Observed relationship (Heaps’ Law):

\[
v = k.n^\beta
\]

where \(v\) is vocabulary size (number of unique words),
\(n\) is the number of words in corpus,
\(k, \beta\) are parameters that vary for each corpus (typical values given are \(10 \leq k \leq 100\) and \(\beta \approx 0.5\)).
Heaps’ Law Predictions

• Predictions for TREC collections are accurate for large numbers of words
  – e.g., first 22,545,922 words of wiki000 scanned
  – prediction is 353,587 unique words
  – actual number is 348,436
• Predictions for small numbers of words (i.e. < 1000) are much worse
Heaps’ Law Predictions

• Heaps’ Law works with very large corpora
  – new words occurring even after seeing 30 million!
• New words come from a variety of sources
  • spelling errors, invented words (e.g. product, company names), code, other languages, email addresses, etc.
• Search engines must deal with these large and growing vocabularies

Stopping

• Function words (determiners, prepositions) have little meaning on their own
• High occurrence frequencies
  – Top 6 words: the, of, and, in, to, a
• Treated as stopwords (i.e. removed)
  – reduce index space, improve response time, improve effectiveness
• Can be important in combinations
  – e.g., “to be or not to be”
Stopping

• Keep track of all very common words in a *stopwords list*.
• During text processing, ignore any word on the list.
• Stopword list can be created from high-frequency words or based on a standard list
• Lists are customized for applications, domains, and even parts of documents
  – e.g., “click” is a good stopword for anchor text

Stopping

• When storage space is not a concern, it can be better to not stop.
  – Queries are less restricted.
  – Remove stop words at query time unless user says to include them.
• Google does not stop.
  – “to be or not to be” returns results.
  – +the returns results (over 14 billion).
End Result of Stopping

• List of words minus those on the stop list.
  – tropical fish include fish found tropical environments around world including both freshwater salt water species fishkeepers often use term tropical fish refer only those requiring fresh water saltwater tropical fish referred marine fish

• Next step: stemming.

Stemming

• Many morphological variations of words
  – inflectional (plurals, tenses)
  – derivational (making verbs nouns etc.)

• In most cases, these have the same or very similar meanings

• Stemmers attempt to reduce morphological variations of words to a common stem
  – usually involves removing suffixes

• Can be done at indexing time or as part of query processing (like stopwords)
Stemming

• Generally a small but significant effectiveness improvement
  – can be crucial for some languages
  – e.g., 5-10% improvement for English, up to 50% in Arabic

<table>
<thead>
<tr>
<th>Arabic</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>kitab</td>
<td>a book</td>
</tr>
<tr>
<td>kitabi</td>
<td>my book</td>
</tr>
<tr>
<td>alkitab</td>
<td>the book</td>
</tr>
<tr>
<td>kitabuki</td>
<td>your book (l)</td>
</tr>
<tr>
<td>kitabuka</td>
<td>your book (m)</td>
</tr>
<tr>
<td>kitabuha</td>
<td>his book</td>
</tr>
<tr>
<td>kataba</td>
<td>to write</td>
</tr>
<tr>
<td>maktaba</td>
<td>library, bookstore</td>
</tr>
<tr>
<td>moktab</td>
<td>office</td>
</tr>
</tbody>
</table>

Words with the Arabic root ktb

Stemming

• Two basic types
  – Dictionary-based: uses lists of related words
  – Algorithmic: uses program to determine related words

• Algorithmic stemmers
  – suffix-s: remove ‘s’ endings assuming plural
    • e.g., cats → cat, lakes → lake
    • Many false negatives: supplies → supplie
    • Some false positives: ups → up
Porter Stemmer

- Algorithmic stemmer used in IR experiments since the 70s
- Consists of a series of rules designed to the longest possible suffix at each step
- Provably effective
- Produces stems not words
- Makes a number of errors and difficult to modify

Porter Stemmer

- Example step (1 of 5)

Step 1a:
- Replace *sses* by *ss* (e.g., stresses → stress).
- Delete *s* if the preceding word part contains a vowel not immediately before the *s* (e.g., gaps → gap but gas → gas).
- Replace *ied* or *ies* by *i* if preceded by more than one letter, otherwise by *ie* (e.g., ties → tie, cries → cri).
- If suffix is *us* or *ss* do nothing (e.g., stress → stress).

Step 1b:
- Replace *ed, edly* by *ee* if it is in the part of the word after the first non-vowel following a vowel (e.g., agreed → agree, feed → feed).
- Delete *ed, edly, ing, ingly* if the preceding word part contains a vowel, and then if the word ends in *at, bl, or tz* add *e* (e.g., fish → fish, pirating → pirate), or if the word ends with a double letter that is not *ll, ss, or zz* remove the last letter (e.g., falling → fall, dripping → drip), or if the word is short, add *e* (e.g., hoping → hope).
- Whew!
Porter Stemmer

<table>
<thead>
<tr>
<th>False positives</th>
<th>False negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>organization/organ</td>
<td>european/europe</td>
</tr>
<tr>
<td>generalization/generic</td>
<td>cylinder/cylindrical</td>
</tr>
<tr>
<td>numerical/numerous</td>
<td>matrices/matrix</td>
</tr>
<tr>
<td>policy/police</td>
<td>urgency/urgent</td>
</tr>
<tr>
<td>university/universe</td>
<td>create/creation</td>
</tr>
<tr>
<td>addition/additive</td>
<td>analysis/analyses</td>
</tr>
<tr>
<td>negligible/negligent</td>
<td>useful/usefully</td>
</tr>
<tr>
<td>execute/executive</td>
<td>noise/noisy</td>
</tr>
<tr>
<td>past/paste</td>
<td>decompose/decomposition</td>
</tr>
<tr>
<td>ignore/ignorant</td>
<td>sparse/sparsity</td>
</tr>
<tr>
<td>special/specialized</td>
<td>resolve/resolution</td>
</tr>
<tr>
<td>head/heading</td>
<td>triangle/triangular</td>
</tr>
</tbody>
</table>

- Porter2 stemmer addresses some of these issues
- Approach has been used with other languages

Krovetz Stemmer

- Hybrid algorithmic-dictionary
  - Word checked in dictionary
    - If present, either left alone or replaced with “exception”
    - If not present, word is checked for suffixes that could be removed
      - After removal, dictionary is checked again

- Produces words not stems
- Comparable effectiveness
- Lower false positive rate, somewhat higher false negative
Stemmer Comparison

Original text:
Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

Porter stemmer:
document describ market strategi carri compani agricultur chemic report predict market share chemic report market statist agrochem pesticid herbicid fungicid insecticid fertil predict sale market share stimul demand price cut volum sale

Krovetz stemmer:
document describe marketing strategy carry company agriculture chemical report prediction market share chemical report market statistic agrochem pesticide herbicide fungicide insecticide fertilizer predict sale stimulate demand price cut volume sale

End Result of Stemming

• List of stemmed terms:
  – tropic fish include fish found tropic environ around world include both freshwat salt water speci fishkeep often use term tropic fish refer onli those requir fresh water saltwat tropic fish refer marin fish
  – (from Porter2 stemmer)

• Next step: advanced processing, or indexing.
Advanced Text Processing

- Part-of-speech tagging.
- Sense disambiguation.
- Synonym classification.
- Named entity tagging.
- Phrase identification.
- Referent resolution.
- Sentence segmentation.
- Translation.
- Speech recognition.

Text Processing Errors

- All text processing is errorful.
  - Design decisions produce segmentation errors, stopping errors, stemming errors.
  - False positives and false negatives.
  - More advanced methods → more difficult processing → more errors.
- Does the benefit outweigh the cost?
  - Segmentation & stemming: definitely.
  - POS tagging, NE tagging: depends on domain.
  - Synonym classes: maybe not.
End Result of Text Processing

- Title: Tropical fish
- Important fields:
  - Links: fish tropic freshwat salt water fishkeep marin fish

Body:
- tropic fish include fish found tropic environ around world include both freshwat salt water speci fishkeep often use term tropic fish refer onli those requir fresh water saltwat tropic fish refer marin fish

Course Project

- Phase I, worksheet 1.
  - Write a text processing module.
  - Parse Wikipedia pages, tokenize, stop, and stem.
  - Answer questions about Wikipedia data: how big is vocabulary, how many word occurrences are there, etc.
- Due next Wednesday.
  - Please start ASAP!
Expectations

- Read Wikipedia pages off disk.
- Identify parts of them that do not need to be indexed.
- Convert the rest into a list of words.
- Drop stop words, stem remaining words to terms.
- Keep track of the number of times each term appears, how many documents it appears in.

PseudoJava

```java
import java.io.*;
import java.util.*;

// HashMap<String, int> termCounts = new HashMap();

File doc = new File(filename);
Scanner docScanner = new Scanner(doc);
while (docScanner.hasNextLine()) {
    List<String> terms = processLine(docScanner.nextLine());
    for (int i=0; i < terms.size(); i++) {
        String currentTerm = terms.get(i);
        int termCount = termCounts.get(currentTerm);
        termCounts.set(currentTerm, termCount+1);
    }
}
docScanner.close()
```
public List processLine(String line) {
    List<String> terms = new List();
    int i = 0;

    Scanner lineScanner = new Scanner(line);
    lineScanner.useDelimiter("\s*");
    while (lineScanner.hasNext()) {
        String word = lineScanner.next();
        /* check if word is appropriate for indexing
         * or if it marks the start of a block to ignore */
        if (word.indexOf("{{") >= 0)
            /* ignore words until closing the block with a }} */
            /* other conditions */
        /* strip non-alphanumeric characters and lower-case */
        word = word.replaceAll("[^a-zA-Z0-9]", "");
        word = word.toLowerCase();
        /* check if word is in the stop list */
        if (!isStopWord(word)) {
            word = stemmer.stem(word); /* stem word */
            terms.set(i, word);
            i++;
        }
    }
    return(terms);
}