Benha University | **Faculty of Computers & Informatics**
Final Exam | **Date**: 2/6 /2013
Class: 4th Year (information System) | **Time:** 3 hours
Subject: Offce Automation Systems | **Examiner:** Dr. Sahar Fawzy

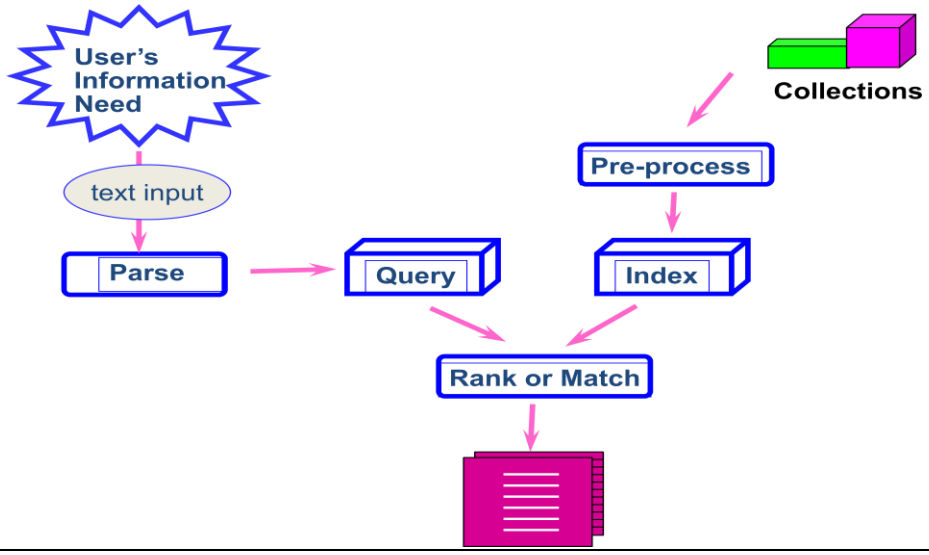
Answer the following (use diagrams when possible):

Question one

1. **"Search is defined as the activity of looking thoroughly in order to find something or someone**" in the sight of this statement, answer the following questions:
   a)What are the main types of search engines?
   There are basically three types of search engines:

1) **Crawler-based search engines (Spiders)** are those that use automated software agents (called crawlers) that visit a Web site, read the information on the actual site, read the site's meta tags and also follow the links that the site connects to performing indexing on all linked Web sites as well. The crawler returns all that information back to a central repository, where the data is indexed. The crawler will periodically return to the sites to check for any information that has changed. The frequency with which this happens is determined by the administrators of the search engine.
2) **Human-powered search engines** rely on humans to submit information that is subsequently indexed and catalogued. Only information that is submitted is put into the index. A human-powered directory, such as the Open Directory, depends on humans for its listings. You submit a short description to the directory for your entire site, or editors write one for sites they review. A search looks for matches only in the descriptions submitted.
   - Changing your web pages has no effect on your listing. Things that are useful for improving a listing with a search engine have nothing to do with improving a listing in a directory. The only exception is that a good site, with good content, might be more likely to get reviewed for free than a poor site.
3) **Hybrid search engine.** In the web's early days, it used to be that a search engine either presented crawler-based results or human-powered listings. Today, it extremely common for both types of results to be presented. Usually, a hybrid search engine will favor one type of listings over another. For example, MSN Search is more likely to present human-powered listings from LookSmart. However, it does also present crawler-based results (as provided by Inktomi), especially for more obscure queries.

b) How search engines work?



c) What's relevance? How to measure the goodness of the retrieved docs?

Relevance information is that suited to your information need.

- *Precision* : Fraction of retrieved docs that are relevant to user's information need.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

- *Recall* : Fraction of relevant docs in collection that are retrieved

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

d) An IR system returns 8 relevant documents, and 10 nonrelevant documents. There are a total of 20 relevant documents in the collection. What is the precision of the system on this search, and what is its recall?

→ Precision = 8/18 = 0.44; Recall = 8/20 =0.4.

1. E) What is the idf of a term that occurs in every document? Compare this with the use of stop word lists.
   It is 0. For a word that occurs in every document, putting it on the stop list has the same effect as idf weighting: the word is ignored.
2. Trace algorithm of phonetic correction for the words "**Robert**", "**Rupert**" and "**Rubin**", comment on the results.
   ➔ Robert and Rupert are the same → R163
   ➔ Rubin is different soundex → R150
3. Illustrate two approaches to supporting the phrase queries. Use examples.

➔ The concept of phrase queries has proven easily understood by users; one of the few "advanced search" ideas that works Many more queries are *implicit phrase queries* For this, it no longer suffices to store only <*term* : *docs*> entries

A first attempt: 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 now a dictionary term
- Longer phrases are processed as we did with wild-cards:
- *stanford university palo alto* can be broken into the Boolean query on biwords: *stanford university AND university palo AND palo alto*

Without the docs, we cannot verify that the docs matching the above Boolean query do contain the phrase.

- False positives, as noted before
- Index blowup due to bigger dictionary
  - Infeasible for more than biwords, big even for them
- Biword indexes are not the standard solution (for all biwords) but can be part of a compound strategy

Solution 2: Positional indexes

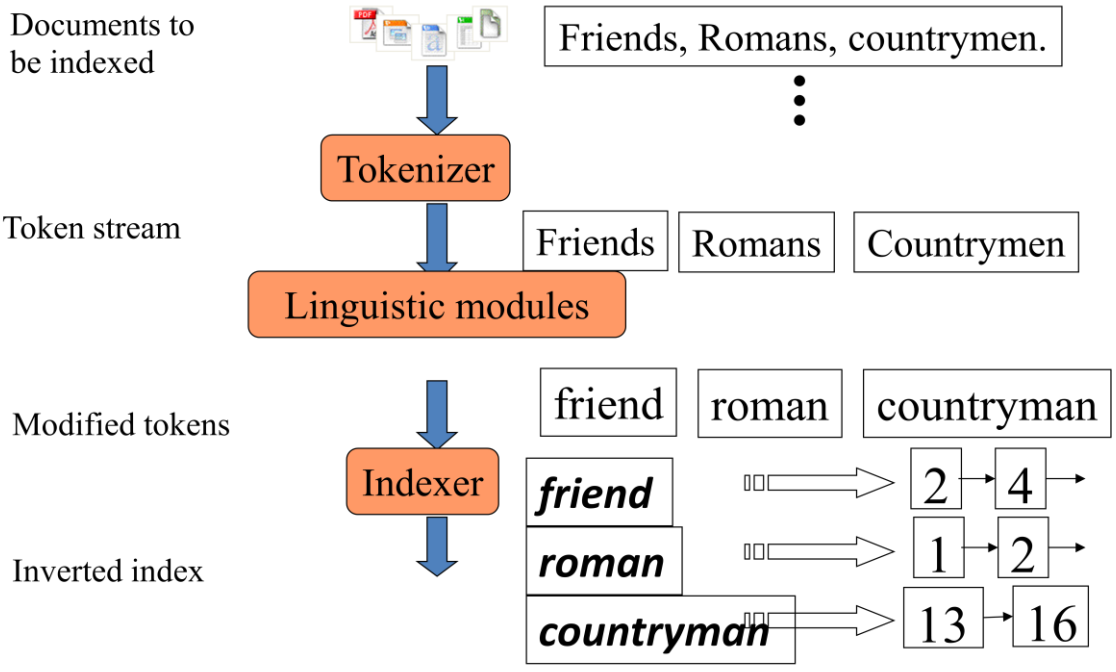
- In the postings, store for each *term* the position(s) in which tokens of it appear:

<*term*, number of docs containing *term*;
doc1: position1, position2 … ;
doc2: position1, position2 … ;
etc.>

- Positional indexes are a more efficient alternative to biword indexes.
- Postings lists in a positional index: each posting is a docID and a list of positions
- Example query: *"to*1 *be*2 *or*3 *not*4 *to*5 *be*6*"*
- <*be*: 993427;
- *1*: 7, 18, 33, 72, 86, 231;
- *2*: 3, 149;
- *4*: 17, 191, 291, 430, 434;
- *5*: 363, 367, …>

Question Two

1. Explain by example the phases of inverted index construction.



2. What is the permuterm index used for? Consider the following document: ***"The universe contains many different universities"***

   a) How many entries a bigram index would contain ? → 50

   b) What is the boolean query on this index for the initial query uni*?

   ➔ **$u and un and ni**

   c) How would you process a query such as univ*, uni*rse, uni*e*se , uni*e*", by using the permuterm index? Give the detail of the processing.

   - **Premuterm index is used to solve the problem of wildcard queries**
   - Univ* → univ*$→ niv*$u→ iv*$un→ v*$uni → $univ*
   - uni*rse → uni*rse$ → ni*rse$u→ i*rse$un → rse$uni*
   - uni*e*s e→ uni* e*se$ → ni*e*se$u → i*e*se$un → *e*se$ uni→ e*se$uni*

---

3. Trace the dynamic programming algorithm for computing the edit distance between strings "**zeil**" and "**trail**". And write down the operations and the cost of each operation.

➔ Edit distance = 3

Input	output	operation	cost
-	t	insert	
z	r	replace	
e	a	replace	
i	I	copy	
l	l	copy	

---

4. Use the 2-gram index and 3-gram index for processing the following wildcard queries
   i. Tol* ii. Rea*
   Is "Tool" result for the wildcard query Tol* ?
   If the answers yes, solve this problem.

➔ 2-gram: $tol* = $t and to and ol && $Rea* = $R and re and ea

➔ 3-gram: $to* = $to and tol && $Rea = $re and rea

➔ Yes tol is a result for the wildcard Tol* when using 2-gram index

➔ To solve this problem, we use post-filter these terms against query.
