Boolean Information Retrieval

Definition of Information Retrieval. Information retrieval is the process of locating one or more resources that satisfy an **information need** and exist within one or more **collections** of resources.

Examples of Information Retrieval.

- finding a webpage that discusses how to clean battery terminals with baking powder
- searching a patent database for recent patents related to information retrieval technology
- recalling all previous sent emails to a friend or colleague
- finding a movie to attend based on one's past ratings of other attended movies
- identifying a cathedral in New York City based on a photograph taken during a vacation in New York
- locating any songs that another song could have perhaps plagiarized

An information retrieval task always begins with a user's **information need**. In other words, what is it that we want to know? We could even go a step further and ask "what is it that we want to feel?". The following is a list of some generic types of information need.

- **Information about people.** Who lost the presidential election of 1968? Who should I contact about obtaining a loan application? Who is favored to win the Best-Actress Academy Award? I want to read a bio of Wayne Gretzky. I want to find a companion to take to next week's concert. Who was the first to postulate the existence of electrical forces? Who is the person in this photograph?
- **Information about places.** Where can I find the nearest Thai restraunt? What city was the capital of the Ottoman Empire? In which county is Alta Dena located? I want to see a map of Santa Clara County. Where does David Letterman reside? Where is the CECS 551 final exam being held? In what city is the tower in this picture located?
- **Information about things.** How many different ant species have been identified? What does it mean to be free? What are the different car models produced by Mazda? What composers have a similar compositional style as the composer who wrote this piece? Is this [photo] an example of poison oak?
- **Information about the time and duration of events.** When did Leif Ericson reach North America? When will the luncheon be held? When will I able to speak with your supervisor? For how long should I bake cod?
- **Information about processes.** How do I get to Dodger Stadium from my house? How can I unclog my sink? How are rainbows formed? How can I make my omlets more fluffy? How to bake salmon? How can I change my state (from sadness, loneliness, anger, etc.)?
- **Information about causality.** Why is my car not starting? Why does the sun's magnetic polarity change every 11 years?

- **Information about comparisons.** Which has fewer calories: bagels or English muffins? What bank is offering the best one-year cd interest rate? Which of these photos best match the one taken by the security camera?
- **Information about quantity.** How many feet are in a single meter? How much would one dollar be worth back in 1965?

An **information retrieval language** is a language (either formal or informal) that is used to specify what resources are to be retrieved. Generally, the user with the information need does not know the structure of each resource, other than possibly some of the text that it might contain. This represents one way in which information retrieval differs from, say, traditional database querying, in which the structure of the database is known to the programmer. For this reason, text informational retrieval languages tend to be **term based**, meaning that the atomic properties are stated with words (i.e. terms) that the resource may or may not possess. Here we assume that each resource contains one or more terms as part of its structure. For example, the resource might be an mp3 music file with terms that provide the song title, artist, date of publication, etc..

In addition to text, some systems allow information to be retrieved via image, sound, or video files. This is referred to as **multimedia information retrieval**. This will be the topic of a later lecture. In this course, however, primary emphasis is placed on text-based information retrieval. This is due to the fact that the vast majority of recorded information and knowledge is text-based; and even the multimedia-based recordings usually are accompanied by text.

For the remainder of the course a **document** is defined as that which represents an individual resource of the collection (also referred to as **corpus**) that is to be searched over. Moreover, we assume that each document contains one or more terms. Thus, there is a many-to-many relationship between terms and documents. And this relationship can be represented by what is called a **term-document matrix**, where each row (respectively column) is represented by a possible term (respectively document). Where entry (t, d) has a 1, iff document d contains term t; and is 0 otherwise. Henceforth we assume both terms and documents can be sorted (e.g., assume that terms have a lexicographical ordering, and each document has a unique id associated with it), and that the matrix rows and columns follow this sorted order.

Example 1. Consider the following corpus of documents, where each document represents a phrase. Assume the set of terms is the union of all English words in each of the phrases. Construct the term-document matrix for this collection. Show how bit-wise operations can be performed on matrix rows to find the documents that have the terms hot `or` cold, and those documents that have the terms hot `and` soup.

- **Document 1:** pea soup hot
- **Document 2:** pea soup cold
- **Document 3:** pea soup in the pot
- **Document 4:** nine days old
- **Document 5:** some like it hot
- **Document 6:** some like it cold

Example 2. A conservative estimate for the Library of Congress holdings is 20 million books. A conservative estimate of the union of all English terms in those books is 500,000. Provide a conservative estimate for the size (in bytes) of the associated Library of Congress term-document matrix.

Boolean query languages. A Boolean query language is an information-retrieval language which, in its simplest form, uses terms as atoms, along with the logical connectives `and`,`or`, and `not`. For example, the Boolean formula

soup and hot

specifies all documents that contain both terms soup and hot. In Boolean logic, a term (or its negation) plays the role of what is called a **literal**; meaning an atomic variable/statement that evaluates to true or false. For example, the term "soup" represents the Boolean literal that translates to "the document contains the term `soup`". Boolean query languages tend to be the languages of choice for ad hoc retrieval tasks. Note the difference, however, between a query formula and the information need that led to the query. For example, if my information need is to know the week for final exams for this semester, then the associated query might be
"csulb fall 2013 final exam schedule".

Henceforth we define a **Boolean information-retrieval system** as one which supports a Boolean query language, and, for each Boolean query q, returns exactly those documents d which satisfy the query; i.e. $q(d)$ evaluates to true. The set of returned documents will henceforth be denoted by $\text{sat}(q)$.

Example 3. Determine $\text{sat}(q)$ for the Example-1 collection, where q is given by

(soup or cold) and not pot

Two measures of effectiveness of a query:

- **Precision.** Denotes the fraction of returned documents that are relevant to the information need.
- **Recall.** Denotes the fraction of documents in the corpus that are relevant to the information need that were returned.

Generally speaking, precision (recall) tends to increase (decrease) as the number of terms used in the (conjunctive) query increases.

The Inverted Index

Given a corpus, an **inverted indexing** of the corpus is represented by the set of terms of the corpus, where each term has a list of references to the documents that contain the term. The collection of terms is called a **dictionary**, while the references are referred to as **postings**.

Example 4. Create an inverted index for the Example-1 corpus.

Given a document collection, let t_i denote the i th most frequently occurring term, measured in terms of the number of documents that it appears in. Then **Zipf's empirical law** states that the fraction f_i of documents that t_i occurs in is given by $f_i = c/i^s$, for some constants $c, s > 0$. This is an example of a **power law**. Intuitively, given a set of possibilities as to what can be observed, a small percentage of those possibilities will be observed a large percentage of the time. This is popularly known as the **Pareto's 80-20 Rule**. For example, "only twenty percent of the dictionary is needed to account for eighty percent of all words that appear in text".

It is interesting to note that it has been empirically verified that the distribution of webpage links also follows a power law. In other words, one can define a directed graph whose vertex set is the set of webpages, and for which there is an edge from one webpage p_1 to another webpage p_2 iff somewhere in p_1 there is a link to p_2. Then the probability density $f(p)$ of webpage p is obtained by dividing the in-degree of vertex p by the total number of edges (links) in the system. Then if we enumerate the pages in descending order of probability, we see that the probabilities descend in accordance with a power law. Larry Page, a Google founder, was among the first to use knowledge of this distribution to better rank the webpages that are returned from a query. In a play on words, he called this **Page Ranking**. Document ranking will be a topic of a later lecture.

Example 5. Assume that the Library of Congress holdings from Example 2 follows Zipf's empirical law with $c = s = 1$. Neglecting the memory needed to store each term, how much memory will be required by an inverted index for this collection? Compare this with the memory needed to store the term-document matrix. Assume that each document id requires 32 bits (4 bytes) of memory.

Another empirical law regarding the recurring frequencies of terms within a collection is Heap's Law which states that the number m of distinct terms counted when processing the first T tokens of text is given by $m = c\sqrt{T}$, where c is a constant that usually ranges between 30 and 100. In other words, as the amount of processed text increases, the rate at which new terms appear approaches zero.

Example 6. Give an approximation for c in Heap's Law for the collection discussed in Example 2, assume that, on average, a Library-of-Congress holding has 100,000 tokens of text.

$T = 20000000 * 100000 =$ number of processed tokens

$m = 500000$ number of terms ... c = 1 / (2 * sqrt(2))

so 100000 is probably too big

How many tokens does the average book have? something more like 40 to 60 thousands... still too big or maybe the law is too simplistic

Efficient Processing of Boolean Queries over an Inverted Index

1. For a conjunctive query, start with the terms that have the fewest postings. A linear scan or binary search can then be performed.
2. Purely conjunctive queries can be performed "in place" within a designated array location.
3. Disjunctive queries can be handled in linear time (in the size of the number of postings) via a merge procedure that is similar to what one uses in the Mergesort algorithm.
4. A negation query can be performed in a manner similar to a conjunctive query. However, now the intersection of the postings is the part that is discarded.

Example 7. Given the postings lists $l_1 = 1, 4, 7, 8, 10, 13, 20, 24, 25, 26, 29$ and
$l_2 = 1, 5, 9, 11, 12, 13, 18, 24, 25, 28, 29, 40, 52,$
determine resulting postings for l_1 and l_2, l_1 or l_2, and l_1 and (not l_2).

(conjuction) if not equal, advance pointer on less than list if equal, mark and advance both pointers once end of list is reached, no more matches and stop search

l_1 intersects l_2 at 1, 13, 24, 25, 29

O(x+y)

(disjunction) l_1 —— l_2 is the merging of the lists and is also O(x+y)

(negation) which is not(l_1) && l_2 same only we're looking for placees where the lists don't match

l_1 or not l_2 doesn't work out very well

Under what conditions is x log y ¡= (x+y) if x ¡¡ y in the worst case?

General Boolean queries. For an arbitrary Boolean query that uses the operators `not`, `and`, and `or`, the query can be converted to disjunctive normal form (DNF) before applying the above efficiency rules. A Boolean formula is said to be in **disjunctive normal form** iff it has the form

$$q_1 \text{ or } q_2 \text{ or } \cdots \text{ or } q_n,$$

where each q_i is a conjunction of literals (i.e. terms, either negated, or occurring positively).

We have "an OR of ANDs"

Example 8. Convert the following Boolean query to an equivalent one that is in DNF form:

```
not(p and not q) and (r or s)
```

(DeMorgans)
(not p or q) and (r or s)
(not p and r) or (not p and s) or (q and r) or (q and s)

Skip Lists. Skip lists are similar to linked lists, but have an extra pointer, called the **skip pointer** that points to a link that occurs further ahead in the list. Skip pointers represent one way of speeding up the processing of a conjunctive query. When processing a binary conjunctive query, at any given time each of the two postings lists has one document id that is under consideration, and the list with the smaller id is the one which is scanned upward from that id until reaching an id that is greater than or equal to the larger id. Having a skip pointer in the scanned list allows one to look ahead to see if this pointer references an id that is less than or equal to the larger id. If so, then one can move directly to that id, instead of having to scan through all the intermediate ids.

Example 9. Given the postings lists $l_1 = 1, 4, 24, 50, 105, 360, 392, 396, 400, 488, 953$ and
$l_2 = 1, 5, 9, 11, 12, 13, 18, 24, 45, 488, 117, 488, 862$,
determine the postings that result in the conjunction of l_1 and l_2, and assume that each id has a skip pointer to an id that is exactly five positions ahead of it (assuming the id is five or more places from the end of the list).

Google Query Operators and Conventions. In the next lecture we will discuss how to achieve some of the functionality described below.

- **Phrase search (“”).** Example: “Todd Ebert” returns pages that have this exact phrase.
- **Search within a specific website (site:).** Example: “Todd Ebert” site:.edu returns pages that have the exact phrase and are located within a .edu domain.
- **Terms to exclude (-).** Example Todd Ebert -CSULB returns pages that have both Todd and Ebert, and not CSULB. The minus sign can also occur before the “site” keyword in order to exclude a website.
- **Wildcard (*).** Example: Super Bowl * Champion returns pages that have the given phrase, with * filled in by one or more terms.
- **Search exactly as is (+).** Similar to minus, but now it prevents the search from including synonyms of the given word. For example, +California history will prevent California from being replaced with an equivalent term, such as ca.
- **The OR operator.** Example Super Bowl 2007 OR 2008 Champion returns pages that will include either 2007 or 2008.

References.

1. A.L. Barabasi, "Linked: How Everything is Connected to Everything Else and What it Means for Business, Science, and Everyday Life", Plume Publishing, 2003

2. Google Guide,

   `www.googleguide.com`

3. C. Manning, P. Raghavan, H. Schutze, "Introduction to Information Retrieval", Cambridge University Press, 2008

4. Wikipedia: Zipf's Law

Exercises.

1. Consider the following document collection:

   - **Doc 1:** breakthrough drug for schizophrenia
   - **Doc 2:** new schizophrenia drug
   - **Doc 3:** new approach to treatment of schizophrenia
   - **Doc 4:** new hope for schizophrenia patients

   Provide both the term-document matrix and inverted index for this document collection. What documents will be returned for the Boolean query `schizophrenia` AND `drug`? For the query `for` AND NOT(`drug` or `approach`)?

2. Suppose that x and y are the respective postings-list sizes for the terms `Romney` and `Obama`. Assuming n documents in the collection. What are the worst-case running times for returning the list of documents that satisfy `Romney` AND NOT `Obama`? `Romney` OR NOT `Obama`? Explain.

3. Convert

   ```
   (a or b) and not(c or d)
   ```

   to disjunctive normal form. If a, b, c, and d are terms with respective postings-list sizes x, y, z, and w, compare the worst-case running times of the original query with the DNF query. Assume n documents in the entire collection.

4. Give an example where a,b, and c are terms with respective postings-list sizes $x < y < z$, and for which

   ```
   (a and b) and c
   ```

will have a longer running time than

```
(c and b) and a
```

5. Use the Integral Theorem to obtain the order of growth of the sequence

$$1 + 1/\sqrt{2} + 1/\sqrt{3} + \cdots .$$

6. Repeat Example 5, but now use $c = 0.5$, followed by $c = 2$. Compare the memory usage for both cases with the $c = 1$ case computed in Example 5.

7. Consider a collection of n documents and m terms, where, for each document d and term t, t appears at most once in d. Moreover, if $t_1, \ldots, t_m$ is a listing of terms in decreasing order of frequency, assume that t_i occurs in $1/i$ of the documents. a) Give a big-Θ estimate of the number of tokens contained in this collection. b) What percentage of the most frequently occurring terms account for 80% of these tokens? Note: Pareto's rule would imply an answer of 20%. Is this correct?

8. For the document collection of the previous problem, how would n have to depend on m in order for Heap's Law to be obeyed. In other words, what should we replace n by in terms of m so that the square root of the number of tokens is on the order of m?

9. Try the following Google queries: burglar, burglar burglar, and burglar OR burglar. Look at the estimated number of results and top hits. Are they identical (as they should be)?

10. Try the Google queries knight, conquer, and knight OR conquer. Are the number of hits for the third query bounded by the sum of the number of hits for the first two queries (as they should be)?