CS630 Lecture 1: Fundamentals of IR
Date: January 26th, 2006
Lecturer: Lillian Lee
Scribes: Ari Rabkin and Victoria Krafft

Administrative note: This lecture largely overlaps with CS430, since that course is not a prerequisite. However, this should not be typical of the course.

Contents

1 Classic Information Retrieval

Information Retrieval is a fairly familiar task to most of us. We all probably use Google at least once a day on average, and are therefore familiar with the classic IR retrieval task: We have some *query*, q, which is an expression of the user's needs. The system compares it with some *corpus* of documents, $C = \{d^{(1)}, \ldots, d^{(n)}\}$. *Corpus* is Latin for body; the corpus in IR is the body of text within which we're trying to find relevant documents.

The goal of the system is to rank (some of) the documents in C by relevance to (the user's needs as expressed by) q.

Typically, IR practitioners will omit the phrase "the user's needs, as expressed by", and just refer to "relevance to a query". Computers cannot read minds, of course, and so the query is generally all the system can use (at least, in the classic, non-relevance-feedback setting). However, it's important to remember that ultimately, the goal is to return things

relevant to the user, not just related to the terms in the query.

2 Methods of Evaluation

Before discussing techniques, we will start by mentioning the techniques we use to evaluate them. A general word of advice: Don't obsess over standard metrics that may not be suitable to our task, but do think about what we're trying to accomplish, and think about how we'd measure it.

2.1 Classic methods

First, we'll look at the "old style" quality measures of classic IR. These measures were common in older work, but newer ones have been developed recently, for reasons we'll discuss in a bit.

To back up for a bit: In information retrieval, we have some corpus C of documents, and some query q. There is some set R of documents in C that are relevant to the query; the information retrieval system returns some other set S.

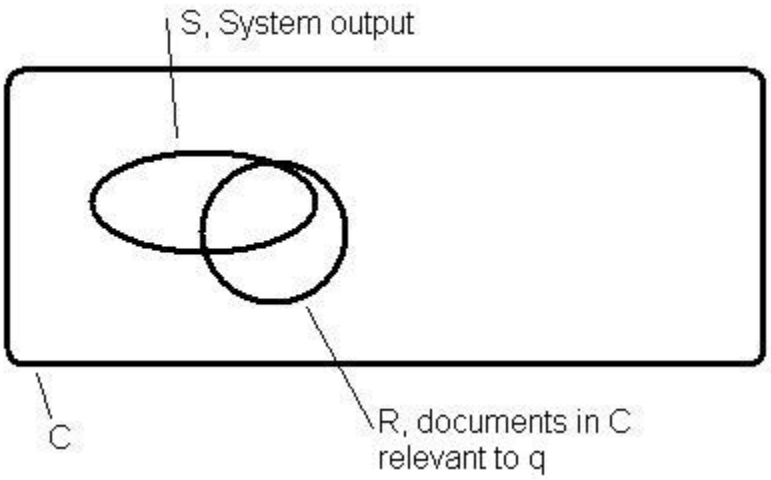

Figure 1: A graphic representing the one retrieval outcome. Ideally, we want the two sets inside C to be the same.

2.1.1 Precision

One thing we could measure is what fraction of the returned documents are relevant: this is called *precision*, and we can write it as $\frac{|S \bigcap R|}{|S|}$. We can also think of this as the likelihood

that a document is relevant, given that the system returns it.

2.1.2 Recall

Alternatively, we could ask about $\frac{|S \bigcap R|}{|R|}$, the fraction of relevant documents that are returned. This is called *recall*.

2.2 Modern Methods

Those are the classic measures of IR, but they're not very useful for a search engine on the web. For most queries on the web, many hundreds or thousands of documents may be relevant, and we probably never look at more than the first few pages of results; we're not interested in whether there are 800 or 1000 possibly-relevant documents returned, if we found what we wanted in the top 20.

Some of this may be an artifact of how web search engines present results, but even so, it's hard to get excited about recall these days. Even precision is a confusing measure in some sense; if the system returns 1000 results, we probably care a great deal about whether the 30 documents that are most relevant are on the first results page, or the 20th.

As a result, it's becoming more common these days to use metrics like "precision at 5", or "precision at 10"; the fraction of the first 5 or 10 documents that are relevant. These are called *high accuracy measures*. Because these more accurately reflect the desired behavior from the system, these measures may be more appropriate these days.

2.3 Data for Experiments

How would we measure precision? We just ask a human "is this document relevant?"! The standard data sets in IR, such as the TREC corpora, include sets of queries, and the documents are coded with relevance judgments with respect to these queries. TREC is a series of annual text retrieval conferences; among other things, these featured "bake-offs" between search engines; the search systems would be tested on these standard corpora. These data sets are widely available, and free to use, and are still standard at SIGIR conferences (SIGIR is the main conference for information retrieval).

3 Vector Space Model

We're trying to find documents with content similar to that expressed by the query. This poses two questions: how to represent content, and how to measure similarity?

The classic techniques use what's called the Vector Space Model (VSM); this is also the basis for modern systems. It was developed largely by Gerry Salton, here at Cornell.

Cornell has a reputation as a theory school, but the VSM is actually highly empirical,and not very theoretical. It is an "ad hoc" approach, where "ad hoc" should be taken in the literal sense: "for this"; in contrast to model-driven approaches, the details of VSM approaches are based largely on intuition and experimentation. It's a simple paradigm, and the computing operations are efficient.

3.1 Representation

The VSM represents documents as vectors in an abstract space. Suppose we have some *vocabulary* $V = \{v^{(1)}, v^{(2)}, \ldots, v^{(m)}\}$, where each $v^{(j)}$ is a content bearing term. Content-bearing terms are not necessarily the same as lexical space-separated words. We might want "data base" to be a single vocabulary term, and we might not want numbers like 4538 to be considered vocabulary terms.

For every document $d \in C$, and for every $v^{(j)} \in V$, we compute a measure of how representative $v^{(j)}$ is for d.

We can then represent d as a document vector $\vec{d} = (d_1, d_2, \ldots, d_m)^T$.

These vectors can be looked at geometrically, as points in $\mathbb{R}^m$. Similar documents will lie near each other, so we can just use closeness in $\mathbb{R}^m$ as our similarity measure.

It may seem routine now, but at the time, it was something of a shock to think of a document, such as Hamlet, as simply a point in space.

We can represent our query q in the same way, as a vector $\vec{q} \in \mathbb{R}^m$, using the terms in q.

3.2 Ranking

There are many possible retrieval schemes. For retrieval, we can measure the match of the document d to the query q by evaluating the inner product of $\vec{d}$ and $\vec{q}$:
$\vec{d} \cdot \vec{q} = \sum_{j=1}^{m} d_j \cdot q_j = \|\vec{d}\|_2 \|\vec{q}\|_2 * cos\cos(\angle(\vec{d} \cdot \vec{q}))$

When ranking documents, the term $\|\vec{q}\|_2$, the length of the query, is the same for all of the matches. It is commonly left out, leaving the score as:
$\|\vec{d}\|_2 * \cos(\angle(\vec{d} \cdot \vec{q}))$

Intuitively, this makes sense, as the cosine term is large if the two document vectors point in the same direction, since the angle between the two vectors is small.

3.3 Term Weights

We haven't yet figured out how to compute the elements d_j, the *term weights* for each document. This is a hard problem, because the ranking is a measure of how relevant term $v^{(j)}$ is to document d. That's not a mathematical concept, and there's no obvious "right way" to approximate a human relevance judgment algorithmically. As a result, many different schemes have been tested, without any clear winner. However, there are three points of broad consensus:

- The importance of terms is corpus-dependent.
  For example, if we have
  q = "computer modeling of neural processes"
  d_1 = "computer chess"
  d_2 = "simulating neural firing"
  Then each of these will produce the same number of matched terms, but if the whole corpus consists of documents about computer science, they should be ranked differently.

- Term frequency is important.
  The number of times that term v_j occurs in d should be taken into consideration. We can use the number of occurrences directly, or take the log, or use a similar mathematical function.

  On the Internet, *web spam* causes problems with this scheme. If the page source has "cheap tickets" in the background 200 times, then unless all d_j's are 0, it will get a higher ranking than a site which only mentions them once, because:
  $\sum q_j d_j \ll \sum q_j 200 d_j$

- Account for document length.
  Simply using the term frequency will cause problems, because a 50,000 word document with 200 occurrences may not be very relevant, while a 10,000 word document with 100 occurrences could be much more useful.

4 Question

This lecture largely covers "classic" information retrieval; it's worth spending a little while to see what's changed, and why it's no longer so applicable.

a) Suppose that we have the query "white house". Suppose that we have a document consisting of nothing but the words "white house". Give a possible term vector for that document, assuming that all term vectors are normalized to length one. If we are not using idf in the weights, what is the similarity between the document and the query?

   Ans: The document vector looks like $\sqrt{2} * (1, 1, 0, \ldots)$ where v_1 = "white" and v_2 = "house". The query vector is identical, and the similarity is $\sqrt{\frac{1}{2}}$.

b) The problem here is that the document "white house", while very similar to the query, is not an interesting or useful document to the vast majority of users. In general, traditional IR doesn't differentiate between useful and useless documents, if they have similar vocabulary. Why wasn't this a problem before the web, and what about the web makes useless documents, and web-spam, an problem?

Ans: Traditional IR was usually assuming that the corpus was assembled from documents that were all guaranteed to be useful for at least some purposes; they were put out as genuine information-bearing documents, such as wire-service stories, technical articles, court decisions, and so forth. Useless documents were simply not put into the corpora. Similarly, the authors of the documents usually were not interested in how a search system would rank their document; there was no incentive to manipulate the rankings.

In contrast, anybody can add a document to the web, and there can be real commercial gain from a high ranking. As a result, people go to some trouble to manipulate the rankings of their page.

c) Most search engines on the Internet use other metrics to help weed out useless documents such as web spam. How could you modify the term weights in a VSM to weed out web spam?

Ans: There are a variety of ways. Some possibilities include:

- Beyond some threshold value for term frequency, either stop increasing the term weight, or drop it. Under this scheme, if web spammers put 1000 instances of "cheap hotel rooms" in the source of the web page, only the first 100 would count. Simply taking the log is not likely to stop the problem; the spammer can use a size 1 font and fit 1000000000 instances in the background instead.
- Increase the term weight only if the terms are visible to the user. If something is hidden in comments, or a white-on-white background, or 1 point font, then don't count it.