CS838-1 Advanced NLP: Information Retrieval

Xiaojin Zhu

2007
Send comments to jerryzhu@cs.wisc.edu

1 Information Retrieval Tasks

Information Retrieval (IR) covers many aspects of getting information. The most well-known task is *ad hoc* retrieval (e.g., Google and Yahoo!), where the task is to find the most relevant documents for a given user query.

- The document collection is known and (roughly) fixed. For example the crawled Web.
- The user issues a query, the IR system ranks documents.
- In some cases, the user can provide *relevance feedback* interactively, e.g. by labeling the top documents as good or bad. The system then updates its ranking based on the label information.
- A related technique is called *pseudo relevance feedback*, where no user interaction is needed. The system treats the top documents as if they were all relevant, and reinforces itself. We will see this again in *semi-supervised learning* later.
- Meta search tries to merge the ranked list from multiple IR systems.
- Clustering and visualizing search results is important too. Try *crane* at `clusty.com`.

Information filtering is another task (e.g., Google Alert). The user defines information need beforehand, and the system picks out relevant documents from a continuous stream. IR extends to images, audio and video too.

2 IR measures

Given a query, a document collection can be divided into 2 parts: those truly relevant and those not (let us assume it is clear cut). An IR system will retrieve

documents it deems relevant, thus dividing the collection into it-thinks-relevant and it-thinks-not. The two divisions are often not the same! Therefore we have four counts:

	truly relevant	truely irrelevant
retrieved	true positive (tp)	false positive (fp)
not retrieved	false negative (fn)	true negative (tn)

Two common measures gauge the quality of the IR system:

- **Precision**
$$P = \frac{tp}{tp + fp}.$$
How pure are the retrieved documents? Obviously we like a precision close to 1.

- **Recall** An IR system can cheat precision by only retrieving the single most confident document. Since the document will very likely be relevant, precision will likely to be 1. However there could be hundreds of truly relevant documents, and a good IR system should get them all. Recall measures this by
$$R = \frac{tp}{tp + fn}.$$
Note recall itself is not a good measure either: the IR system can cheat by returning the whole document collection.

For this reason precision and recall is almost always reported as a pair.

An IR system usually has many tunable parameters. Different parameter settings result in different P, R value pairs (assuming a fixed document collection). One can plot these pairs on a *precision-recall curve*, where by convention x-axis is recall and y-axis is precision. Note the parameters are hidden and not shown. PR curve is often used to compare two IR systems: the one with curve above (and to the right) is superior.

A single number summary of precison and recall is the *F-score*

$$F = \frac{2PR}{P + R},$$

which is the harmonic mean of P, R.

There are many other measures, but P, R, F form the basis.

3 Language Models for IR

There are many ways to do IR. LM is one of them. Given a query q, we want to rank the documents. We introduce a binary label r for 'relevant', and compute the probability

$$p(r = 1|q, d)$$

that a given document d is relevant to the query q. Ranking by this probability gives what we want. Applying Bayes rule

$$p(r=1|q,d) = \frac{p(d,q|r=1)p(r=1)}{p(q,d)}. \tag{1}$$

It will be convenient to consider the *log odds ratio*, which preserves the ranking:

$$\begin{aligned}
& \log \frac{p(r=1|q,d)}{p(r=0|q,d)} && (2)\\
= & \log \frac{p(d,q|r=1)p(r=1)}{p(d,q|r=0)p(r=0)} && (3)\\
= & \log \frac{p(q|d,r=1)p(d|r=1)p(r=1)}{p(q|d,r=0)p(d|r=0)p(r=0)} && (4)\\
= & \log \frac{p(q|d,r=1)p(r=1|d)}{p(q|d,r=0)p(r=0|d)} && (5)\\
= & \log \frac{p(q|d,r=1)}{p(q|d,r=0)} + \log \frac{p(r=1|d)}{p(r=0|d)} && (6)
\end{aligned}$$

We make the assumption that

$$p(q|d,r=1)$$

is computed from a (document) unigram language model created from d, and applied to q. That is, we assume the query is "generated" by the relevant document. Note there will be many separate document LMs.

We make a second assumption that

$$p(q|d,r=0) = p(q|r=0),$$

which says since d is irrelevant, q, d are conditionally independent. It can be ignored for ranking.

The term $\log \frac{p(r=1|d)}{p(r=0|d)}$ measures how likely a document d is relevant, without even knowing what query will be asked! It can be viewed as an a priori importance measure. It can be a constant for all documents, but a more interesting model made Google famous, as we discuss later.

4 Vector Models for IR

Each document d is represented by a $tf \cdot idf$ vector (many other variations exist). tf is *normalize term frequency*, i.e. word count vector scaled, so that the most frequent word type in this document has a value 1:

$$tf_w = \frac{c(w,d)}{\max_v c(v,d)},$$

where $c(w, d)$ is the count of word w in d. *idf* is inverse document frequency. Let N be the number of documents in the collection. Let n_w be the number of documents in which word type w appears.

$$idf_w = \log \frac{N}{n_w}.$$

idf is an attempt to handle stopwords or near-stopwords: if a word (like "the") appears in most documents, it is probably not very interesting. Finally

$$tf \cdot idf_w = tf_w \times idf_w.$$

The query q is represented by a $tf \cdot idf$ vector too. Documents are ranked by their *cosine similarity* to q: q and a document d form an angle θ in the $tf \cdot idf$ vector space, and

$$\begin{aligned} sim(d, q) &= \cos(\theta) && (7) \\ &= \frac{d^\top q}{\|d\| \cdot \|q\|} && (8) \\ &= \frac{d^\top q}{\sqrt{d^\top d}\sqrt{q^\top q}}, && (9) \end{aligned}$$

where the dot product

$$u^\top v = \sum_{i=1}^{V} u_i v_i.$$