CS 630 Lecture 1: 01/26/2006

Lecturer: Lillian Lee Scribes: Asif-ul Haque, Benyah Shaparenko

---

This lecture focuses on the following topics

- Information retrieval fundamentals
- Vector Space Model (VSM)
- Deriving term weights in VSM

1 Information retrieval fundamentals

1.1 "Classic" retrieval setting

In the "classic" information retrieval setting, there is a query q which is an the expression of user's needs. There is also a "corpus" (body) of documents

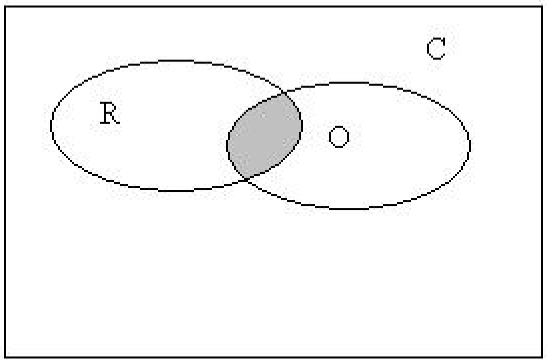
$$C = \{d^{(1)}, d^{(2)}, \ldots, d^{(n)}\}$$

The goal is to rank (some of) the documents in C by relevance to [the user's need as indicated by] q.

1.2 Quality measures

So we need some quality measures as well to perform evaluation. Given a query q, suppose a system outputs the set O of documents, and let the set of documents that are relevant to the query q be R. There are two well known measures.

- **Precision** answers what proportion of the output documents is relevant to the query. Mathematically, precision is computed by
$$\frac{|O \cap R|}{|O|}$$
- **Recall** answers what proportion of the relevant documents was obtained by the system. Mathematically,
$$\frac{|O \cap R|}{|R|}$$



In modern systems and document collections, there are thousands of relevant documents for a query. So $|R|$ can, conservatively, range from 1k to 10k while the number of output documents that are reviewed is only $10 - 20$. Thus the value of **recall** is really small. For this reason **recall** not appropriate. The notion of precision can be changed somewhat as well to get a "high accuracy measure." **precision @** 10, for example, asks how many of the top 10 output documents are relevant to the query.

1.3 Measuring precision

How do we know which documents are relevant to a given query? We can do it in 2 ways.

- We can get a lot of data from the "Text REtrieval Conference" (TREC) database or similarly, relevance-annotated collections, which we can use as "evaluation corpora" to know the relevance.
- We can perform relevance judgments in our system.

2 Vector Space Model

The goal of the system is to get documents whose **content** is **similar** to that expressed by q. So we have two issues here.

1. How to represent the content of a document
2. How to compute the similarity

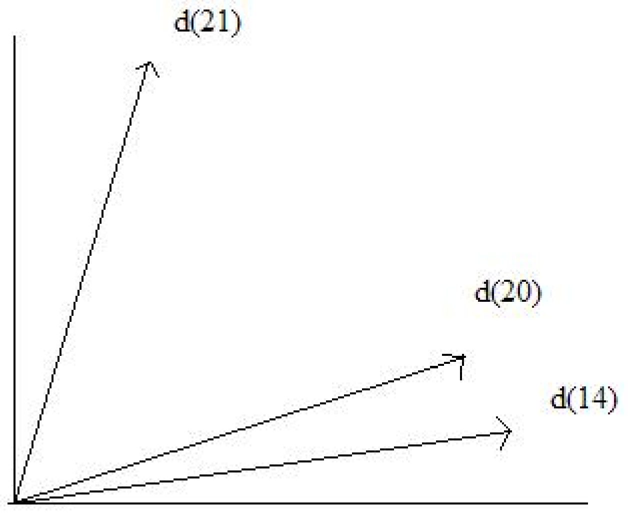
The basis for modern systems is the Vector Space Model (VSM). Gerry Salton of the CS department here was the pioneer of this empirical ("ad-hoc") approach compared to model-driven approaches.

2.1 Representation

Let V be the vocabulary of content-bearing terms.

$$V = \{v^{(1)}, v^{(2)}, \dots, v^{(m)}\}$$

A term can have multiple words; for example, in "data base," the two words in the phrase hold some meaning collectively. A term can also be a sequence of digits like "34539753671" to express some meaning like phone number, ISBN number, etc. For every document $d \in C$ and for every $v^{(j)} \in V$ we compute a weight for how representative $v^{(j)}$ is of d's context. So for every document d we get a document vector $\overrightarrow{d} = (d_1, d_2, \ldots, d_m)^T$. For example if $v^{(1)}$="dogs" and $v^{(2)}$="cats," some document $\overrightarrow{d}^{(14)} = (8, 1.2)^T$ means that the document's relevance measure with "dogs" is 8 and with "cats" is 1.2. Similarly the vectors $\overrightarrow{d}^{(20)} = (7, 2)^T$ and $\overrightarrow{d}^{(21)} = (1, 6)^T$ are representations of two other documents. So each document is a vector in $\Re^m$, and we can measure the "similarity" by the closeness of the vectors in space. Note that $d^{(20)}$ and $d^{(14)}$ are clearly closer to each other than they are to $d^{(21)}$.



2.2 Retrieval

There are many different possibilities for score functions for performing ranked retrieval, but they all tend to measure the "match" between $\vec{d}$ and $\vec{q}$. The similarity here can be measured by the inner product

$$\vec{d} \cdot \vec{q} = \sum_{j=1}^{m} d_j q_j = \|\vec{d}\|_2 \|\vec{q}\|_2 \cos(\angle(\vec{d}, \vec{q}))$$

which is true if $\|\vec{d}\|_2, \|\vec{q}\|_2 > 0$ and where d_j represents whether term j is good for document d and q_j represents whether term j is good for the query. A "0" value for the norm could be because of an empty query or document, but could also result from an out-of-vocabulary document.

Since $\|\vec{q}\|_2$ is identical for all documents, the $\|\vec{q}\|_2$ term is irrelevant with respect to ranking. So we have that

$$\text{Match Score} \overset{\text{rank}}{=} \|\vec{d}\|_2 \cos(\angle(\vec{d}, \vec{q}))$$

Intuitively, the cos term asks whether $\vec{q}$ and $\vec{d}$ point in the same direction. The $\|\vec{d}\|_2$ term captures:

1. whether the document has lots of words

2. very representative terms, i.e. "dogmaticity" (which is probably not a word)

This setup is the retrieval paradigm that we will use. But, where are all the d_i terms in these vectors coming from?

3 Term-weighting question

Recall several definitions. The notation d_j is how much $v^{(j)}$ represents d's content. Ideally, somebody would read the documents and create weights for the terms, but if you are a computer and cannot read, then what?

The goal is vague, so there isn't just one right way to come up with term weights. However, by many years of experience, there is a consensus about what should be included when doing term weighting. In fact, there are three important factors for the weighting function.

3.1 Representativeness is corpus-dependent.

Some terms are more important than other terms, and their representativeness depends on the context of those terms. For example, given two terms, "the" and "computer," the term "computer" seems to be more important, but if all the documents in the corpus at hand have the term "computer," then when comparing documents, it has no use whatsoever.

3.1.1 An example

Let q = "computer modeling of neural processes." Let $d^{(1)}$ = "computer chess" and $d^{(2)}$ = "simulating neural firing." Since q shares one term in common with each $d^{(1)}$ and $d^{(2)}$, then $d^{(1)}$ and $d^{(2)}$ would be considered equally relevant if we only considered term overlap. However, this doesn't make sense. Although there are the same number of matches of terms, the documents should be ranked differently. In fact, $d^{(2)}$ is quite relevant even though it happens to have used different terms, namely, "simulating" for "modeling" and "firing" for "processes." So, we need to factor the rarity of the term $v^{(j)}$ into the weights across the corpus.

Note that if $d^{(1)}$ were the *only* doc in the corpus that mentioned computers, then it might be possible that $d^{(1)}$ is highly relevant, in that the user might be specifically querying a non-computer-science collection looking for an application of computers.

3.1.2 IDF

One way of measuring the rarity of a term is by dividing by the number of documents containing $v^{(j)}$ to get something called "inverse document frequency". This will henceforth be called simply idf. It is based on the idea of dividing by some quantity related to the number of documents that contain $v^{(j)}$ For example,

$$\text{idf could be } \frac{\#\text{ docs in } C}{\#\text{ docs containing } v^{(j)}}$$

If the term occurs in all the documents, idf = 1. If the term occurs in only one document, then idf = $|C|$. Alternatively, we could take the log of all the idf scores, which serves the purpose of compressing the dynamic range of the idf function.

3.2 Representation of a particular document

The term frequency of $v^{(j)}$ in d is also important. The intuition is that the more often a term appears, the more "content-full" that term is. Ways of incorporating term frequency include using $\#(v^{(j)}$ in $d)$, $\log \#(v^{(j)}$ in $d)$, among others.

Is term frequency always the right thing to do? In the case of "web spam," certain web sites have lots of hidden text written in the same color as the background, so that when you view the page, it looks normal, but all the words are repeated, say, 200 times. When performing retrieval, this document gets ranked higher because of the inflated term frequency score.

$$\sum q_j d_j \ll \sum q_j (200 d_j)$$

if we assume that not all the d_j's are zero.

3.3 Normalization

We can't just use only term frequency, but must perform some kind of normalization on the term weights. Next time, we will begin with cosine normalization, then talk about pivoted length normalization.

4 Finger Exercise: What are terms?

In class, we briefly discussed what should be terms. In particular, it is generally not safe to assume that each word corresponds to a term. This finger exercise illustrates how term selection can affect retrieval results.

Consider the following problem. There are two documents in the corpus:

$d^{(1)}$ = ithaca 's weather is rainy
$d^{(2)}$ = a student studying in the department of computer science at cornell in ithaca

Assume that we have a query and we want to retrieve the *single* document that best matches the query. The query is the following:

q = the weather in ithaca

1. If we assume that each word is one term, what is the set of terms? Using the TFIDF weighting (without logs) discussed in class with cosine normalization, which document will be retrieved?

2. Assume that we have the following list of *stopwords*: the, in, is, a, of, at. (Stopwords are words we choose to omit from the vocabulary as being non-content-bearing.) In this case, what is the set of terms? Using the TFIDF weighting discussed in class with cosine normalization, which document will be retrieved?

3. What about stemming? (Stemming is one way of consolidating words that have similar content and form. For example, the words “stem,” “stemmed,” “stemming,” “stems,” “stemmer,” etc. could all be condensed into one word, “stem.”) Could it have a similar effect? How much difference do you think stopword removal, stemming, or combining words, e.g. “computer science” instead of “computer” and “science” actually have in practice? (Since this question is open-ended, there is no solution given.)

5 Solution

1. This is the list of terms

Term ID	idf	Term
1	1	ithaca
2	2	's
3	2	weather
4	2	is
5	2	rainy
6	2	a
7	2	student
8	2	studying
9	2	in
10	2	the
11	2	department
12	2	of
13	2	computer
14	2	science
15	2	at
16	2	cornell

The query is expressed (unnormalized) as:

$$q = (1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0)^T$$

The idf function used here is:

$$\text{idf} = \frac{\#\text{ docs in } C}{\#\text{ docs containing } v^{(j)}}$$

The documents are represented as:

$$d^{(1)} = \frac{1}{\sqrt{17}}(1, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T$$

and

$$d^{(2)} = \frac{1}{\sqrt{57}}(1, 0, 0, 0, 0, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2)^T$$

In this case, $d^{(2)}$ is selected because

$$q \cdot d^{(1)} = \frac{1}{\sqrt{17}}(1 + 2) \approx 0.7276$$

$$q \cdot d^{(2)} = \frac{1}{\sqrt{57}}(1 + 4 + 2) \approx 0.9272$$

2. The documents and query are now the following:

$d^{(1)}$ = ithaca 's weather rainy
$d^{(2)}$ = student studying department computer science cornell ithaca
q = weather ithaca

This is the list of terms

Term ID	idf	Term
1	1	ithaca
2	2	's
3	2	weather
4	2	rainy
5	2	student
6	2	studying
7	2	department
8	2	computer
9	2	science
10	2	cornell

The query is expressed (unnormalized) as:

$$q = (1, 0, 1, 0, 0, 0, 0, 0, 0, 0)^T$$

The documents are represented as:

$$d^{(1)} = \frac{1}{\sqrt{13}}(1, 2, 2, 2, 0, 0, 0, 0, 0, 0)^T$$

and

$$d^{(2)} = \frac{1}{5}(1, 0, 0, 0, 2, 2, 2, 2, 2, 2)^T$$

Now, $d^{(1)}$ is now selected because

$$q \cdot d^{(1)} = \frac{1}{\sqrt{13}}(1 + 2) \approx 0.8321$$

$$q \cdot d^{(2)} = \frac{1}{5}(1) = .2$$