Inverted Indexes: The Basics

Yufei Tao

KAIST

March 26, 2013

We already know that, given a query with a set Q of terms, a search engine computes a score for every webpage D, denoted as $score(D, Q)$. Then, all the webpages are sorted in descending order of their scores, after which the top k webpages are returned, where k is a parameter chosen by the search engine (e.g., 100).

Computing the $score(q, D)$ for every D, however, is expensive, and may easily take a few hours. So, what is it that a search engine relies on in order to return the query result in almost real time?

The answer is the inverted index, as is the topic of this lecture.

Let us pave the way for our subsequent discussion by defining the query result precisely. Let Q be the sequence of query terms. Recall that, in the space vector model, every document D (a.k.a. webpage, in our context) is converted to a point $p = (p[1], ..., p[d])$, where d is the dimensionality equal to the size of our dictionary $DICT$. Similarly, Q is also converted to a point $q = (q[1], ..., q[d])$. We define the score of D as:

$$score(D, Q) \quad = \quad \frac{\sum_{i=1}^{d}(p[i] \cdot q[i])}{|p| \cdot |q|} \cdot rank(D).$$

where $rank(D)$ is the page rank of D.

Note

As mentioned before, the score function used by a search engine (e.g., Google) is typically kept as a commercial secret. Nevertheless, the above definition is good enough for us to discuss many central ideas behind inverted indexes.

Further recall that, if we denote w_i as the i-th $(1 \le i \le d)$ term in $DICT$, then

$$\begin{aligned} p[i] &= tf(D, w_i) \cdot idf(w_i) \\ q[i] &= tf(Q, w_i) \cdot idf(w_i) \end{aligned}$$

where $tf(D, w_i)$ is the term frequency of w_i in D (similarly for $tf(Q, w_i)$), and $idf(w_i)$ is the inverse document frequency of w_i. Hence, we can re-write $score(D, Q)$ as:

$$score(D, Q) = \frac{\sum_{i=1}^{d} tf(D, w_i) \cdot tf(Q, w_i) \cdot idf(w_i)^2}{|p| \cdot |q|} \cdot rank(D).$$

Observe that, in the score formula of the previous slide, the terms $rank(D)$ and $|p|$ depend only on the document D itself, but not on the query. Hence, if we define

$$A(D) = rank(D)/|p|$$

then the formula can be simplified into:

$$score(D, Q) = \frac{A(D)}{|q|} \cdot \sum_{i=1}^{d} tf(D, w_i) \cdot tf(Q, w_i) \cdot idf(w_i)^2$$

For simplicity, let us consider that every term appears at most once in Q (which is true for most queries in practice anyway). As a result, $tf(Q, w_i) = 1$ if $w_i \in Q$, and 0 otherwise. Hence, if we denote $Q = \{t_1, t_2, ..., t_m\}$, where $m = |Q|$, then we can further simplify the score formula into:

$$score(D, Q) = \frac{A(D)}{|q|} \cdot \sum_{t_i \in Q} tf(D, t_i) \cdot idf(t_i)^2$$

Finally, let us forget about $|q|$ because it is the same for all documents, and hence, does not affect the ordering of their scores. This leads to our final score definition:

$$score(D, Q) = A(D) \cdot \sum_{t_i \in Q} tf(D, t_i) \cdot idf(t_i)^2$$

Now we can finally state the problem to be solved by inverted indexes. Let $S = \{D_1, ..., D_n\}$ be a set of documents (i.e., webpages). Given a query set Q, we want to report the k documents with the largest scores, where the score of a document is calculated as in the previous slide.

The following fact should have become obvious:

Lemma

If a document D does not contain any term in Q, then $score(D, Q) = 0$.

Motivated by this, let us process the query by focusing on the terms $t_1, ..., t_m$ in Q. In particular, we want to know what are the documents containing t_1? And, respectively, $t_2, ..., t_m$? In fact, why don't we pre-compute this information to avoid generating it at query time?

This is exactly the idea of inverted indexes.

Inverted Index

An inverted index consists of:

- For every term w_i in $DICT$, the value of $idf(w_i)$.
- For every term w_i in $DICT$, an inverted list, denoted as $list(w_i)$, which contains a pair
$$(i, tf(D_i, w_i))$$
for every document D_i that contains w_i.

We will refer to i as the *document id* of D_i.

Example (Excerpted from [Zobel and Moffat, 2006]

Suppose that our document collection is:

document ID	content
1	the old night keeper keeps the keep in the town
2	in the big old gown in the big old house
3	the house in the town had the big old keep
4	where the old night keeper never did sleep
5	the night keeper keeps the keep in the night
6	and keeps in the dark and sleeps in the light

Example (Excerpted from [Zobel and Moffat, 2006])

term w	inverted list for w
and	(6, 2)
big	(2, 2), (3, 1)
dark	(6, 1)
did	(4, 1)
gown	(2, 1)
had	(3, 1)
house	(2, 1), (3, 1)
in	(1, 1), (2, 2), (3, 1), (5, 1), (6, 2)
keep	(1, 1), (3, 1), (5, 1)
keeper	(1, 1), (4, 1), (5, 1)
keeps	(1, 1), (5, 1), (6, 1)
light	(6, 1)
never	(4, 1)
night	(1, 1), (4, 1), (5, 2)
old	(1, 1), (2, 2), (3, 1), (4, 1)
sleep	(4, 1)
sleeps	(6, 1)
the	(1, 3), (2, 2), (3, 3), (4, 1), (5, 3), (6, 2)
town	(1, 1), (3, 1)
where	(4, 1)

Think

How would you construct all the inverted lists from a set of documents?

The Overall Index

In general, given a set S of documents $D_1, ..., D_n$, we create:

1. An inverted index on S.
2. An array $A = (A(D_1), A(D_2), ..., A(D_n))$.
    - See Slide 5 for the definition of $A(D)$.

The above provide all the necessary information for answering a query, as shown in the next slide.

Query Algorithm

A query with term set $Q = \{t_1, ..., t_m\}$ can be answered as follows:

```
algorithm query($Q$)
1. $score(D, Q) = 0$ for all $D \in S$
2. for each term $t_j \in Q$
3. for each pair $(i, tf(D_i, t_j))$ in $list(t_j)$
4.     $score(D_i, Q) = score(D_i, Q) + tf(D_i, t_j) \cdot idf(t_j)^2$
5. for each $D \in S$
6. $score(D, Q) = score(D, Q)/A(D)$
7. sort the documents by score
8. return the $k$ documents with the highest scores
```

Think

Why does the above algorithm return the correct result?