Inverted Indexes: Trading Precision for Efficiency

Yufei Tao

KAIST

April 1, 2013

After compression, an inverted index is often small enough to fit in memory. This benefits query processing because it avoids disk I/Os. In this lecture, we will discuss several additional tricks to make a query algorithm even faster. As we will see, some of these tricks trade efficiency for precision, namely, they do not guarantee returning the same results as our current algorithm, but the differences are usually minor in practice, and hardly affect the usefulness of a search engine.

First, let us recall the score definition we have been using to judge the importance of a webpage D with respect to a query Q that consists of terms $t_1, ..., t_m$:

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

where

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

where p is the point converted from D.

One effective heuristic in practice is to modify the score function to

$$score^*(D, Q) \quad = \quad A(D) \cdot \sum_{t_j \in Q} \max\{trim(tf(D, t_j) \cdot idf(t_j)^2, \tau)\}$$

where τ is a system parameter and function $trim(x, \tau)$ is defined as:

$$trim(x, \tau) \quad = \quad \begin{cases} 0 & \text{if } x < \tau \\ x & \text{otherwise} \end{cases}$$

The intuition is that, if the contribution $tf(D, t_i) \cdot idf(t_j)^2$ of a term t_j is too small, we may ignore it without harming the quality of the query result significantly.

The new score definition has an important implication: for each term t_j, it suffices to consider only those pairs $(i, tf(D_i, t_j))$ in its inverted list such that

$$tf(D_i, t_j) \quad \geq \quad \tau / idf(t_j)^2$$

Hence, we can sort the pairs $(i, tf(D_i, t_j))$ of an inverted list in descending order of $tf(D_i, t_j)$, so that in query processing we can scan the list from the beginning, and stop as soon as hitting a pair that does not satisfy the above inequality.

See the next slide for an example.

Example (Adapted 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	(2, 2), (6, 2), (1, 1), (3, 1), (5, 1),
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	(5, 2), (1, 1), (4, 1)
old	(1, 1), (2, 2), (3, 1), (4, 1)
sleep	(4, 1)
sleeps	(6, 1)
the	(1, 3), (3, 3), (5, 3), (2, 2), (6, 2), (4, 1)
town	(1, 1), (3, 1)
where	(4, 1)

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 $(i, tf(D_i, t_j))$ of $list(t_j)$ in descending order of $tf(D_i, t_j)$
4.          if $tf(D_i, t_j) > \tau / idf(t_j)^2$ then
5.              $score^*(D_i, Q) = score^*(D_i, Q) + tf(D_i, t_j) \cdot idf(t_j)^2$
6.          else break
7.  for each $D \in S$
8.      $score^*(D, Q) = score^*(D, Q) / A(D)$
9.  sort the documents by score
10. return the $k$ documents with the highest scores
```

Compression

Since the sorting method has changed, we also need to modify the way we compress an inverted list.

Recall that, previously, the pairs $(i, tf(D_i, t))$ of the inverted list of term t are sorted in ascending order of document id i. Now, they are first sorted by frequency $tf(D_i, t)$, and then, by document id i. For example:

the | (1, 3), (3, 3), (5, 3), (2, 2), (6, 2), (4, 1)

Let us refer to the sequence of pairs with the same frequency as a segment. For example, (1, 3), (3, 3), (5, 3) form a segment, (2, 2), (6, 2) form another, and so does (4, 1).

Compression (cont.)

In general, a segment $(f, id_1), (f, id_2), ..., (f, id_x)$ is stored as:

$$(f, id_1, id_2 - id_1, id_3 - id_2, ..., id_x - id_{x-1})$$

where each number is represented as Elia's gamma or delta code.

For example, the inverted list of the previous slide is stored as:

the | (3, 1, 2, 2), (2, 2, 4), (1, 4)

Remarks

So far we have learned two types of inverted indexes:

- Id-sorted: where the pairs of an inverted list are sorted by document id.
- Frequency-sorted: where the pairs of an inverted list are sorted by term frequency.

Both types of inverted indexes are collectively referred to as document-level inverted indexes.