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:

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

where \( \text{rank}(D) \) 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 ($1 \leq i \leq d$) term in $DICT$, then

$$p[i] = tf(D, w_i) \cdot idf(w_i)$$
$$q[i] = tf(Q, w_i) \cdot idf(w_i)$$

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:

$$score(D, Q) = \sum_{i=1}^{d} \frac{tf(D, w_i) \cdot tf(Q, w_i) \cdot idf(w_i)^2}{|p| \cdot |q|} \cdot rank(D).$$
Observe that, in the score formula of the previous slide, the terms $\text{rank}(D)$ and $|p|$ depend only on the document $D$ itself, but not on the query. Hence, if we define

$$A(D) = \frac{\text{rank}(D)}{|p|}$$

then the formula can be simplified into:

$$\text{score}(D, Q) = \frac{A(D)}{|q|} \cdot \sum_{i=1}^{d} tf(D, w_i) \cdot tf(Q, w_i) \cdot \text{idf}(w_i)^2$$
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 $\text{score}(D, Q) = 0$.

Motivated by this, let us process the query by focusing on the terms $t_1, \ldots, t_m$ in $Q$. In particular, we want to know what are the documents containing $t_1$? And, respectively, $t_2, \ldots, t_m$? 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.
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 $D_i$. 
Suppose that our document collection is:

<table>
<thead>
<tr>
<th>document ID</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the old night keeper keeps the keep in the town</td>
</tr>
<tr>
<td>2</td>
<td>in the big old gown in the big old house</td>
</tr>
<tr>
<td>3</td>
<td>the house in the town had the big old keep</td>
</tr>
<tr>
<td>4</td>
<td>where the old night keeper never did sleep</td>
</tr>
<tr>
<td>5</td>
<td>the night keeper keeps the keep in the night</td>
</tr>
<tr>
<td>6</td>
<td>and keeps in the dark and sleeps in the light</td>
</tr>
</tbody>
</table>
### Inverted Indexes: The Basics

<table>
<thead>
<tr>
<th>term $w$</th>
<th>inverted list for $w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>(6, 2)</td>
</tr>
<tr>
<td>big</td>
<td>(2, 2), (3, 1)</td>
</tr>
<tr>
<td>dark</td>
<td>(6, 1)</td>
</tr>
<tr>
<td>did</td>
<td>(4, 1)</td>
</tr>
<tr>
<td>gown</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>had</td>
<td>(3, 1)</td>
</tr>
<tr>
<td>house</td>
<td>(2, 1), (3, 1)</td>
</tr>
<tr>
<td>in</td>
<td>(1, 1), (2, 2), (3, 1), (5, 1), (6, 2)</td>
</tr>
<tr>
<td>keep</td>
<td>(1, 1), (3, 1), (5, 1)</td>
</tr>
<tr>
<td>keeper</td>
<td>(1, 1), (4, 1), (5, 1)</td>
</tr>
<tr>
<td>keeps</td>
<td>(1, 1), (5, 1), (6, 1)</td>
</tr>
<tr>
<td>light</td>
<td>(6, 1)</td>
</tr>
<tr>
<td>never</td>
<td>(4, 1)</td>
</tr>
<tr>
<td>night</td>
<td>(1, 1), (4, 1), (5, 2)</td>
</tr>
<tr>
<td>old</td>
<td>(1, 1), (2, 2), (3, 1), (4, 1)</td>
</tr>
<tr>
<td>sleep</td>
<td>(4, 1)</td>
</tr>
<tr>
<td>sleeps</td>
<td>(6, 1)</td>
</tr>
<tr>
<td>the</td>
<td>(1, 3), (2, 2), (3, 3), (4, 1), (5, 3), (6, 2)</td>
</tr>
<tr>
<td>town</td>
<td>(1, 1), (3, 1)</td>
</tr>
<tr>
<td>where</td>
<td>(4, 1)</td>
</tr>
</tbody>
</table>
Think

How would you construct all the inverted lists from a set of documents?
In general, given a set $S$ of documents $D_1, \ldots, D_n$, we create:

1. An inverted index on $S$.
2. An array $A = (A(D_1), A(D_2), \ldots, A(D_n))$.
   - 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.
A query with term set $Q = \{t_1, \ldots, t_m\}$ 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?