Simplified Phrase Search Algorithm

Kai Eckert
University of Mannheim
kai@informatik.uni-mannheim.de

November 11, 2008

1 Algorithm

This is a simplified version of the POSITIONAL-INTERSECT algorithm presented by Manning et al. in Section 2.4.2, Figure 2.12 [1].

```
FORWARD-POSITIONAL-INTERSECT($p_1, p_2, k$)
answer $\leftarrow \langle\rangle$
while $p_1 \neq$ NIL and $p_2 \neq$ NIL
    do if docId($p_1$) = docId($p_2$)
        then $pp_1 \leftarrow$ positions($p_1$)
            $pp_2 \leftarrow$ positions($p_2$)
            while $pp_1 \neq$ NIL and $pp_2 \neq$ NIL
                do if pos($pp_2$) − pos($pp_1$) = $k$
                    then ADD(answer, docId($p_1$), pos($pp_1$))
                        $pp_1 \leftarrow$ next($pp_1$)
                        $pp_2 \leftarrow$ next($pp_2$)
                elseif pos($pp_2$) − pos($pp_1$) > $k$
                    then
                        $pp_1 \leftarrow$ next($pp_1$)
                else
                        $pp_2 \leftarrow$ next($pp_2$)
            $p_1 \leftarrow$ next($p_1$)
            $p_2 \leftarrow$ next($p_2$)
    elseif docId($p_1$) > docId($p_2$)
        then $p_2 \leftarrow$ next($p_2$)
    else
            $p_1 \leftarrow$ next($p_1$)
    return answer
```

p1, *p2*, *pp1* and *pp2* are pointers to lists. *p1* and *p2* reference the document lists of the two terms to be intersected. *pp1* and *pp2* reference the inner position lists for each document.

docId and pos dereference the pointers to their actual value in the list, i.e. a document identifier or a position. positions extracts the inner position list from an entry in the document list.

ADD adds a document identifier and a position to the resulting document list.

The simplification arises from the specialization on one direction of the nearby operator and that the distance must match exactly, that means, that documents are included only, if term 2 is found *exactly* k terms *after* term 1.

This simplified version is sufficient to support common phrase searches, as shown in the following algorithm.

```
PHRASE-SEARCH($index, phrase$)
1  ▷ $index$ is a positional index, according to example below
2  $t \leftarrow$ TERMS($phrase$)
3  $k \leftarrow 1$
4  $answer \leftarrow$ INDEX-GET($index, t$)
5  $t \leftarrow$ next($t$)
6  while $t \neq$ NIL and $answer \neq \langle\rangle$
7        do $nextDocs \leftarrow$ INDEX-GET($index, t$)
8           $answer \leftarrow$ FORWARD-POSITIONAL-INTERSECT($answer, nextDocs, k$)
9           $k \leftarrow k + 1$
10          $t \leftarrow$ next($t$)
11 return $answer$
```

Here, TERMS splits the phrase in a list of tokens and normalizes them, in the same manner like the indexed text was processed before during index creation.

INDEX-GET returns the document list for a single term t from the *index*.

2 Positional Index

The positional index contains for each term a list of documents and for each of these documents a list of positions indicating where the term is found.

For example, consider the following format:

- term: ⟨ doc1: ⟨pos1, pos2, ...⟩l, doc2: ⟨pos1, pos2, ...⟩⟩; etc.

In this format, a positional index can look like this:

- fools: ⟨ 2: ⟨1,17,74,222⟩, 4: ⟨8,78,108,458⟩, 7: ⟨3,13,23,193⟩⟩;
- in: ⟨ 2: ⟨3,37,76,444,851⟩, 4: ⟨10,20,110,470,500⟩, 7: ⟨5,15,25,195⟩⟩;
- rush: ⟨ 2: ⟨2,66,194,321,702⟩, 4: ⟨9,69,149,429,569⟩, 7: ⟨4,14,404⟩⟩;

3 Remark

This simplified phrase search approach was developed for an exercise in a digital libraries lecture. If you find any mistakes or weaknesses or you have some improvements, please contact me via email.

References

[1] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, July 2008.