Counting strongest term occurrence matches first

Paul Elschot

26 March 2017

Abstract

This report describes an implementation of scoring term occurrences for nested proximity queries. Counting the strongest matches first allows to reuse existing implementations of ranking models. Term occurrence scores need more investigation.

Contents

Author email: `paul.j.elschot@gmail.com`

1 Term occurrence scores

In many ranking formulas the score for a document D for bag of words query is a sum over n term (or word) scores $s_i(t_f)$. These term scores depend among others on the term frequency t_f in the document:

$$S(D,Q) = \sum_{i=1}^{n} s_i(t_f) \tag{1}$$

Here the dependencies of t_f on D and i are omitted for brevity. For example in the BM25 formula [9] the score for a term in a document is:

$$s_i(t_f) = IDF(q_i)\frac{t_f(k_1+1)}{t_f + k_1(1-b+b\frac{|D|}{avgdl})} \tag{2}$$

Here q_i is the query term, $IDF(q_i)$ is the inverse document frequency weighting factor for the term, $|D|$ is the length of the document, *avgdl* is the average length of the documents in the collection, and k_1 and b are parameters.

The author is not aware of any report on the use of term occurrence scores u_{ij} that sum up to the term score in a document:

$$s_i(t_f) = \sum_{j=1}^{t_f} u_{ij} \tag{3}$$

In most ranking formulas, including (2), the score value for a term in a document decreases with the document length $|D|$ and increases ever slower with the term frequency in the document t_f. Therefore it can be assumed that the score for a single term occurrence depends on all positions of the occurrences of this term in the document. It might also be expected that when two occurrences of the same term are closer to each other, they will each have a lower score. Taking this one step further, it might be possible to improve the document score for a term by using the positions of the term occurrences for bag of words queries, without combining the positions of occurrences of different terms. The absolute position of a term is also of influence on the term occurrence score, see for example [5].

One possible choice for the term occurrence scores u_{ij} that is independent of the term positions relates directly to the term frequency:

$$u_{ij} = s_i(j) - s_i(j-1), \text{ with } s_i(0) = 0 \tag{4}$$

In most ranking formulas these values become smaller with increasing j to express a diminishing influence of an extra occurrence of the same term. Another position independent choice for u_{ij} could be the average over m_f matching terms:

$$u_{ij} = \frac{s_i(m_f)}{m_f} \tag{5}$$

Since implementations of $s_i(t_f)$ are available, both (4) and (5) are easily implementable. Position dependent choices for u_{ij} will be possible, but these are not readily available.

2 Weights from nested proximity queries

2.1 Matching term occurrences

When there is a match for a proximity query, the document score can depend on the matching distance via a distance kernel k_Q for the query Q [7, 10]. The values of such kernels normally vary between 1 and 0 for increasing distances. Some examples are a Gaussian kernel (6) and an inverse distance kernel (7). Using p_1 and p_2 as matching positions in D these are:

$$k_Q(p_1, p_2) = exp[\frac{1 - (p_1 - p_2)^2}{2\sigma_Q^2}] \tag{6}$$

$$k_Q(p_1, p_2) = \frac{1}{|p_1 - p_2|} \tag{7}$$

Here it is assumed that the matching distance $d = |p_1 - p_2|$ is always at least 1. In many cases this kernel only depends on the matching distance, so for simplity $k_Q(d)$ will be used instead of $k_Q(p_1, p_2)$.

When nested proximity queries express concept dependencies [2] recommends to use a longer dependency range than for dependencies between terms. For example in the Gaussian kernel (6) a larger σ_Q could be considered in such cases.

When evaluating nested proximity queries, the kernel values can be multiplied at each query nesting to provide a nested kernel value w_p at each matching term position:

$$w_p = \prod_Q k_Q(d), \text{ for matching occurrences} \tag{8}$$

Here the dependencies of d on Q and on the actual matching position are omitted for brevity. At a matching term position w_p may not be unique when the term at this position matches the query more than once. In such cases the maximum available value can be used.

For BM25, an alternative to the use of distance kernels is the use of virtual regions in [3]. Such virtual regions are produced by query operators from term positions. For such regions the problem that the same occurrence of a term within a document can be counted more than once is solved by calibrating the weights for the query operators.

2.2 Non matching term occurrences

In [8] it was found that for a mix of term queries and proximity queries over the same terms, the term queries should have a significant impact on the results. In [4] there is a constraint that adding one query term to a document must increase the score of a bag of words query. Therefore non matching term occurrences should have an influence on the document score. The distance kernel value $k_Q(f)$ of a non matching (far) distance f can be used for this:

$$w_p = \prod_Q k_Q(f), \text{ for non matching occurrences} \tag{9}$$

When the query tree has multiple leaves, the minimum w_p at the leaves can be used. For each (sub)query Q, the far distance f should be chosen as at least the largest allowed proximity query distance. Good choices for the far distance values will also depend on the distance kernels and on the collection of documents.

3 Weighted term occurrence scores

To reuse the well developed bag of words ranking formulas for proximity queries, the sum of the term occurrence scores (3) will be weighted (10) to provide a weighted document score (11) instead of (1):

$$\bar{s}_i(t_f) = \sum_{j=1}^{t_f} w_{ij} u_{ij} \tag{10}$$

$$\bar{S}(D, Q) = \sum_{i=1}^{n} \bar{s}_i(t_f) \tag{11}$$

For the choices of the occurrence scores u_{ij} from equation (4) or (5) and of w_{ij} from the kernel values w_p the following constraints are considered:

- when all term occurrences match at the shortest possible distance, the document score should be the same as for a bag of words query that does not require proximity,
- an extra occurrence of a query same term should result in a higher document score [4],
- a query that allows a higher distance between the same terms should should never have a lower score, and
- when a term occurrence matches, it should score higher than when it does not match.

The following choices provide scoring consistency for these constraints, as illustrated in appendix A. For the occurrence scores u_{ij} the term frequency related one (4) is chosen, and for the weights w_{ij} the kernel values w_p from (8) and (9) are chosen so that the strongest matches of the position query are counted first:

$$w_{ij} = \{w_p \text{ for term } i \text{ in non increasing order}\}_j \tag{12}$$

4 Implementation

An implementation of equations (4), (8), (9), (10), (11) and (12) is available for the Lucene search engine. This implementation allows only a single distance kernel, and query weights can be multiplied into (8) for (nested) queries and into (10) for terms. The non matching distance f in (9) can be provided for each (sub)query Q. The default value for f is large so that its kernel value is very small and non matching term occurrence scores do not influence the weighted document score.

Equation (4), $u_{ij} = s_i(j) - s_i(j-1)$, is implemented by evaluating $s_i(j)$ for each matching term occurrence, and this is expected to have the biggest influence on performance. Other than BM25, in Lucene 6.0 [1] implementations for $s_i(j)$ can be chosen from divergence from randomness models, information based models and divergence from independence models.

The main runtime complexity factor for this implementation is the sorting of the term occurrence weights w_{ij} for equation (12). This complexity is $O(m_f log(m_f))$ per term per document, where m_f is the number of matching terms in the document.

Table 1: Implementation overview

Function	Added and changed classes in package `org.apache.lucene.search.spans`
wrapper for existing `SpanQuery`	`SpansTreeQuery`
non matching term occurrence weight, equation (9)	`SpansTreeWeight`
equation (11)	`SpansTreeScorer`
equations (4), (10) and (12)	`AsSingleTermSpansDocScorer`
nested distance kernel values, equation (8)	`ConjunctionNearSpansDocScorer` `DisjunctionNearSpansDocScorer`
far distance, section 2.2	`SpanNearQuery`
disjunction proximity, section 4.1	`SpanOrQuery` `DisjunctionNearSpans` `DisjunctionNearSpansDocScorer`
synonym terms, section 4.2	`SpanSynonymQuery` `SynonymSpans` `SynonymSpansDocScorer`

For more details on the implementation [6] see Table 1 and the documentation comments in the program code.

4.1 Maximum scoring distance for disjunctions

The nested kernel values of equation (8) are also used in a disjunction query that uses the distances between the matches of the subqueries. When only one subquery is present, or when the subquery matches are too far apart, a maximum scoring distance is used. This disjunction query can be used as a bag of concepts query when it is preferred that the concepts occur close to each other.

4.2 Synonym terms

Equation (4) has been implemented for a group of terms, using the term with the highest document frequency from the group for the weight. This term has the lowest IDF in (2). The total term frequency of the group in a document is used as the document term frequency t_f.

5 Further work

Retrieval experiments with this scoring model have not yet been done. Such experiments normally have a document collection, a choice of topics, and the relevant documents for these topics. Proximity queries for these topics will have to be created for such experiments.

Nested proximity queries and counting strongest term occurrence matches first may need longer distance kernels k_Q. The weight for the non matching term occurrences will also need investigation.

To have scoring consistency for term occurrences there may be other ways than counting the strongest matches first. The values w_p in (8) are available at each matching position, and if term occurrence scores were also available at each term position, scoring could done by using the position order for the weights and for the term occurrence scores in (10).

References

[1] Lucene PMC Apache Foundation. Lucene 6.0 Similarity package. `https://lucene.apache.org/core/6_0_0/core/index.html?org/apache/lucene/search/similarities/Similarity.html`.

[2] Michael Bendersky and W Bruce Croft. Modeling higher-order term dependencies in information retrieval using query hypergraphs. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 941–950. ACM, 2012.

[3] Roi Blanco and Paolo Boldi. Extending BM25 with multiple query operators. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 921–930. ACM, 2012.

[4] Hui Fang and ChengXiang Zhai. An exploration of axiomatic approaches to information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 480–487. ACM, 2005.

[5] Laurie Hirsch. Evolved Apache Lucene SpanFirst queries are good text classifiers. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[6] Paul Elschot (issue reporter). Spans tree scoring. `https://issues.apache.org/jira/browse/LUCENE-7580`, 2016 (accessed March 2017).

[7] Yuanhua Lv and ChengXiang Zhai. Positional language models for information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306. ACM, 2009.

[8] Donald Metzler and W Bruce Croft. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM, 2005.

[9] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

[10] Tao Tao and ChengXiang Zhai. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 295–302. ACM, 2007.

Table 2: Example documents and queries.

$D1$	`t1`	`t2`	.	.	.
$D2$	`t1`	.	.	`t2`	`t1`
$Q1$	`t1` near `t2`, maximum distance 1				
$Q2$	`t1` near `t2`, maximum distance 3				

Table 3: Term `t1` scores $\bar{s}_1$ for $Q1$ in $D2$, for different choices of u_{1j} and w_{1j}.

u_{1j}, w_{1j}	$\bar{s}_1$	$\bar{s}_1$ for $k(f) \to 0$
u_{1j} average (5)	$(1 + k(f))s_1(2)/2$	$s_1(2)/2$
u_{1j} from (4), w_{1j} doc order	$k(f)s_1(1)+(s_1(2)-s_1(1))$	$s_1(2) - s_1(1)$
u_{1j} from (4), w_{1j} non increasing (12)	$s_1(1)+k(f)(s_1(2)-s_1(1))$	$s_1(1)$

A Scoring consistency

To illustrate that the choices of equations (4) and (12) can result in consistent term score values, consider the example documents and queries in Table 2. The documents are of equal length, so the term occurrence scores (4) do not depend on the document. In both documents `t2` occurs once at distance 1 from `t1`, so `t2` always has the same score value $\bar{s}_2(1) = s_2(1)$.

In the second document `t1` also occurs at distance 3 from `t2`. Query $Q2$ illustrates that for `t2` this results in two weights w_p, and the larger one of these, the weight for the shortest matching distance, is chosen.

A first constraint for scoring consistency is that when all term occurrences match at the shortest possible distance, the document score is the same as for a bag of words query that does not require proximity. This is the trivial case of the document score of $D1$ for both $Q1$ and $Q2$. Since the weights $w_{11} = w_{21} = k(1) = 1$ the document score $\bar{S}(D1, Q1) = S(D1, Q1) = s_1(1) + s_2(1)$.

A second scoring constraint is that an extra occurrence of the same term should result in a higher document score. For this consistency it is necessary that the term score of `t1` in $D2$ is bigger than in $D1$ because `t1` occurs more often in $D2$ and the documents have equal lengths. In $D1$ `t1` occurs once, and using $t_f = 1$ in either (4) or (5), the term occurrence score is $u_{11} = s_1(1)$.

The middle column of Table 3 shows term scores for `t1` in $D2$ for some choices of the term occurrence scores and of the weights, and in the right column this is compared to $s_1(1)$, the term score in $D1$ for very small kernel distance values.

In $D2$ `t1` occurs twice, so the term occurrence scores u_{11} and u_{12} must be determined. When the average of (5) is used, as illustrated in the first row of Table 3, the score for `t1` in $D2$ can be lower than its score in $D1$, because the series in (4) is decreasing with j and distance kernel values $k(d)$ for distances larger than 1 can be arbitrarily close to 0:

$$s_1(2)/2 < s_1(1)$$

Therefore the term frequency related score of (4) is used.

This leaves the term occurrence weights w_{1j} are to be chosen. For $Q1$ the first, non matching, occurrence of `t1` in $D2$ has weight $w_p = k(f)$, the other occurrence has weight $w_p = k(1) = 1$. When using the document order, as illustrated in the second row of Table 3, the same inconsistency can occur:

$$s_1(2) - s_1(1) < s_1(1)$$

Therefore the non increasing order of (12) is used.

A third scoring constraint is that a query that allows a higher distance between the same terms should should never have a lower score. This is achieved because a higher distance always involves at least the same matching term occurrences with the same weights and when more term occurrences match the score increases. The score increases because the weight for matching term occurrences is not smaller than the minimum possible product of the kernel values for the maximum allowed distances, which is chosen as the weight for non matching occurrences.

A fourth constraint is that when a term occurrence matches it should score higher than when it does not match. This depends on the choice of the non matching distance f, and this choice is left to the user.