中科汇联问答系统
lucene 检索性能的提高

刘　荣
北京中科汇联信息技术有限公司
2014-12-11

目录

- **lucene search**
  1. TFIDF Similarity
  2. problem of search multi-filed
  3. improvement in search
  4. z-mert optimization

Lucene 检索--TFIDF Similarity

- Lucene Conceptual Scoring Formula

$$\text{score}(q,d) = \text{coord-factor}(q,d) \cdot \text{query-boost}(q) \cdot \frac{V(q) \cdot V(d)}{|V(q)|} \cdot \text{doc-len-norm}(d) \cdot \text{doc-boost}(d)$$

- Description for Formula

1. Query-boost for the query (actually for each query term) is known when search starts.

2. Query Euclidean norm |*V(q)*| , There are two good reasons to keep this normalization.

2.1 Recall that Cosine Similarity can be used find how similar two documents are.

2.2 Applying query normalization on the scores helps to keep the scores around the unit vector, hence preventing loss of score data because of floating point precision limitations.

Lucene 检索--TFIDF Similarity

- Lucene Practical Scoring Formula

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left(\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot \text{t.getBoost}() \cdot \text{norm}(t,d) \right)$$

- Description for Formula

1. tf(t in d) $\quad \text{tf}(t \text{ in } d) = \text{frequency}^{½}$
2. idf(t)

$$\text{idf}(t) = 1 + \log\left(\frac{\text{numDocs}}{\text{docFreq}+1} \right)$$

3. coord(q,d)

   coord(q,d) is a score factor based on how many of the query terms are found in the specified document.

4. t.getBoost()

   t.getBoost() is a search time boost of term *t* in the query *q* .like “jakarta^4 apache”,4 is the term boost

5. norm(t,d) :encapsulates a few (indexing time) boost and length factors

$$\text{norm}(t,d) = \text{lengthNorm} \cdot \prod_{\text{field } f \text{ in } d \text{ named as } t} f.\text{boost}()$$

Lucene 检索--TFIDF Similarity

- Lucene Practical Scoring Formula

$$\text{score}(q,d) = \text{coord}(q,d) \cdot \text{queryNorm}(q) \cdot \sum_{t \text{ in } q} \left(\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot t.\text{getBoost}() \cdot \text{norm}(t,d) \right)$$

- Description for Formula

6. queryNorm

$$\text{queryNorm}(q) = \text{queryNorm}(\text{sumOfSquaredWeights}) = \frac{1}{\text{sumOfSquaredWeights}^{1/2}}$$

$$\text{sumOfSquaredWeights} = q.\text{getBoost}()^2 \cdot \sum_{t \text{ in } q} \left(\text{idf}(t) \cdot t.\text{getBoost}() \right)^2$$

Lucene 检索-- multi field search

- Multi-field search

  1. query search(booleanQuery)

     Q= q1 or q2 or q3. which q1 is searching field1,q2 is searching field2…

     q1= term1 or term2 or term3.

- Problem in multi-field search

  1. queryNorm calculated

$$\mathrm{q1}(sumOfSquaredWeights) = q1.getBoost()^2 \times \sum_{t\ in\ q1}(idf(t) * t.getBoost)^2$$

$$Q(sumOfSquaredWeights) = \sum_{q\ in\ Q} q(sumOfSquaredWeights)$$

$$queryNorm(Q) = \frac{1}{\sqrt{Q(sumOfSquaredWeights)}}$$

$$queryNorm(q1) = queryNorm(Q) \times q1.getBoost$$

so the queryNorm for every single query is not independent . When some field contain unusual word that doesn’t match key word. It is influence the result.

Lucene 检索-- multi field search

- Multi-field search

  1. query search(booleanQuery)

     Q= q1 or q2 or q3. which q1 is searching field1,q2 is searching field2…

     q1= term1 or term2 or term3.

- Problem in multi-field search

  1. queryNorm calculated

  2. coord calculated

     coord(q1,field)= max_contain_word/query_word.

     max_contain_word : how many of the query terms are found in the specified document

     query_word : how many of query terms

     coord(Q) = max_contain_query/query_number

     max_contain_query : how many of the query that score is not zero.

     query_number : how many of the query

  so it influence the result if it have some field that is not important.

Lucene 检索-- MERT

- MERT

1. maximum score

$$\text{score}(e_k, f) = \sum_{m=1}^{M} \lambda_m \phi_m(e_k, f).$$

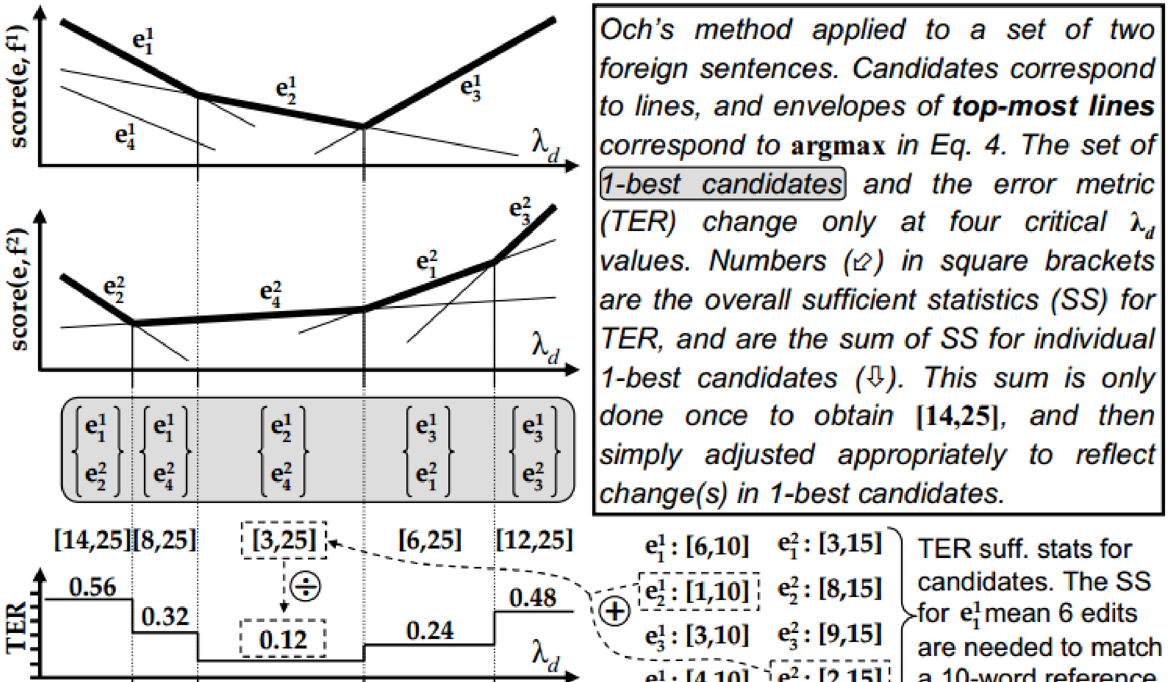
where f-foreign sentence , canditate set for f be $\{e_1, e_2, .. e_K\}$. feature vector $\phi(e,f)$=$\{\phi_1(e,f) \ldots \phi_M(e,f)\}$

2. parameter estimation using Och's Method

2.1 fix the d^{th} dimension

$$\text{score}(e_k, f) = \lambda_d \phi_d(e_k, f) + \sum \lambda \phi (e, f)$$

2.2 vary λ_d



Lucene 检索-- MERT

- MERT

  2. parameter estimation using Och's Method

     2.1 fix the d^{th} dimension

     $$\text{score}(\mathrm{e}_k, f) = \lambda_d \phi_d(\mathrm{e}_k, f) + \sum_{m \neq d} \lambda_m \phi_m(\mathrm{e}_k, f).$$

     2.2 vary λ_d

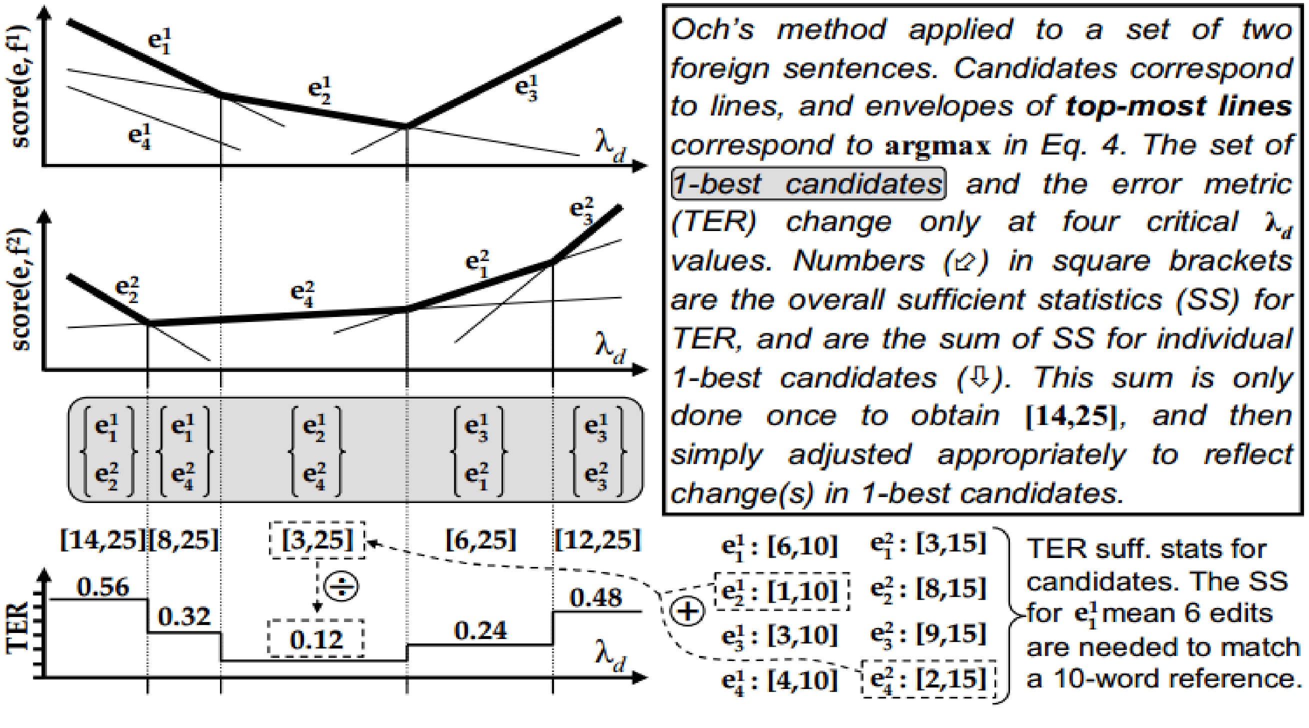


Figure 1. Och's method applied to a set of two foreign sentences.

Lucene 检索 -- MERT

- MERT

  2. parameter estimation using Och's Method

…timization (line 17) the efficient method of Och (2003).

Input: Initial weight vector $\Lambda^0 = \{\Lambda^0[1], \ldots, \Lambda^0[M]\}$; `numInter`, the number of initial points per iteration; and N, the size of the candidate list generated each iteration.

Return: Final weight vector $\Lambda^* = \{\Lambda^*[1], \ldots, \Lambda^*[M]\}$.

1. Initialize $\Lambda \leftarrow \Lambda^0$
2. Initialize $\texttt{currError} \leftarrow +\infty$
3. Intialize the cumulative candidate set for each sentence to the empty set.
4. **loop**
5.   Using Λ, produce an N-best candidate list for each sentence, and merge it with the cumulative candidate set for that sentence.
6.   **if** no candidate set grew **then** Return Λ // MERT convergence; we are done.
7.
8.   Initialize $\Lambda_1 \leftarrow \Lambda$
9.   **for** ($j = 2$ to `numInter`), initialize $\Lambda_j \leftarrow$ random weight vector
10.
11.   Initialize $j_{\texttt{best}} \leftarrow 0$
12.   **for** ($j = 1$ to `numInter`) **do**
13.   Initialize $\texttt{currError}_j \leftarrow \mathrm{error}(\Lambda_j)$ based on cumulative candidate sets
14.   **repeat**
15.   Initialize $m_{\texttt{best}} \leftarrow 0$
16.   **for** ($m = 1$ to M) **do**
17.   Set $(\lambda, \texttt{err})$ = value returned by efficient investigation of the m^{th} dimension and the error at that value (i.e. using Och's method)
18.   **if** ($\texttt{err} < \texttt{currError}_j$) **then**
19. $m_{\texttt{best}} \leftarrow m$
20. $\lambda_{\texttt{best}} \leftarrow \lambda$
21. $\texttt{currError}_j \leftarrow \texttt{err}$
22.   **end if**
23.   **end for**
24.   **if** ($m_{\texttt{best}} \neq 0$) **then**
25.   Change $\Lambda_j[m_{\texttt{best}}]$ to $\lambda_{\texttt{best}}$
26.   **end if**
27.   **until** ($m_{\texttt{best}} == 0$)
28.   **if** ($\texttt{currError}_j < \texttt{currError}$) **then**
29. $\texttt{currError} \leftarrow \texttt{currError}_j$
30. $j_{\texttt{best}} \leftarrow j$
31. $\Lambda \leftarrow \Lambda_j$
32.   **end if**
33.   **end for**
34.   **if** ($j_{\texttt{best}} == 0$) **then** Return Λ // Could not improve any further; we are done.
35. **end loop**

Lucene 检索-- MERT

- MERT

3. lucene optimization with MERT

二 步骤 :

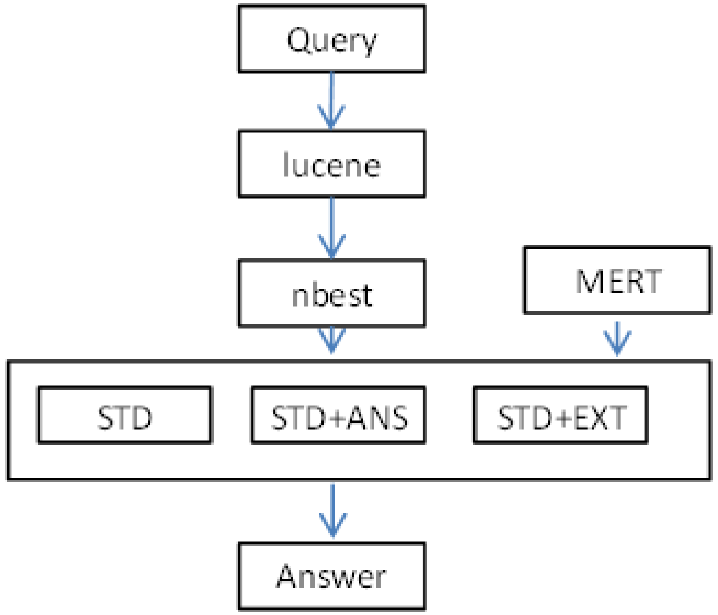
1. 对于 query 经过 lucene 搜索后产生 nbest（取 5），为：q1,q2,q3,q4.q5。

2. nbest rescore:

循环遍历 nbest

2.1 $score(q_1, query) = w_1 \times f_1(q_1, query, STD) + w_{2x} f_1(q_1, query, STD + ANS) + w_3 \times f_1(q_1, query, STD + EXT)$

重新排序选出 TOP1



Reference

- Lucene4.0 http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html
- F. **Och**. 2003. Minimum Error Rate Training in Statistical Machine Translation. In Proceedings of ACL, pages 160-167.
- **Zaidan**. 2009. Z-MERT: A Fully Configurable Open Source Tool for Minimum Error Rate Training of Machine Translation Systems. The Prague Bulletin of Mathematical Linguistics, No. 91:79-88.