Improving Document Ranking for Long Queries with Nested Query Segmentation

Rishiraj Saha Roy[1]*, Anusha Suresh[2], Niloy Ganguly[2], and Monojit Choudhury[3]

[1] Max Planck Institute for Informatics, Saarbrücken, `rishiraj@mpi-inf.mpg.de`
[2] Indian Institute of Technology (IIT), Kharagpur, {`anusha.suresh, niloy`}`@cse.iitkgp.ernet.in`
[3] Microsoft Research India, Bangalore, `monojitc@microsoft.com`

Abstract. In this research, we explore nested or hierarchical query segmentation[4], where segments are defined recursively as consisting of contiguous sequences of segments or query words, as a more effective representation of a query. We design a lightweight and unsupervised nested segmentation scheme, and propose how to use the tree arising out of the nested representation of a query to improve ranking performance. We show that nested segmentation can lead to significant gains over state-of-the-art flat segmentation strategies.

1 Introduction

Query segmentation [1–5] is one of the first steps towards query understanding where complex queries are partitioned into semantically coherent word sequences. Past research [1–5] has shown that segmentation can potentially lead to better IR performance. Till date, almost all the works on query segmentation have dealt with *flat* or *non-hierarchical segmentations*, such as: `windows xp home edition | hd video | playback`, where pipes (`|`) represent flat segment boundaries. For short queries of up to three or four words, such flat segmentation may suffice. However, slightly longer queries of about five to ten words are increasing over the years ($\simeq 27\%$ in our May 2010 Bing Australia log) and present a challenge to the search engine. One of the shortcomings of flat segmentation is that it fails to capture the relationships between segments which can provide important information towards the document ranking strategy, particularly in the case of a long query.

These relationships can be discovered if we allow nesting of segments inside bigger segments. For instance, instead of a flat segmentation, our running example query could be more meaningfully represented as Fig. 1. Here, the *atomic segments* – `windows xp` and `hd video`, are progressively joined with other words to produce larger segments – `windows xp home`, `windows xp home edition`, and `hd video playback`. It is intuitive from this representation that `windows`

* This research was completed while the author was at IIT Kharagpur.
[4] An extended version of this paper is available at `www.abc.com`

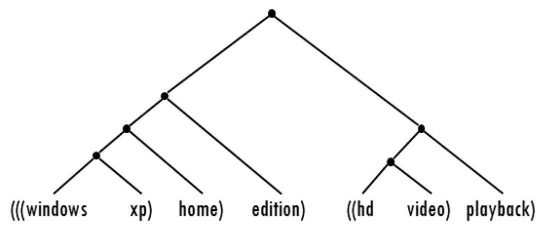

Fig. 1. Nested segmentation tree.

`xp` and `hd video` are non-negotiable (atomic units) when it comes to matching within documents, and the strength of ties between word pairs can be said to weaken as they move farther in terms of the (unique) path through the tree. We define some of the important concepts below.

Tree distance. The tree distance[5] $td(t_1, t_2; n(q))$ between two terms t_1 and t_2 in $n(q)$, the nested segmentation of a query q, is defined as the shortest path (i.e., the number of hops) between t_1 and t_2 (or vice versa) through the nested segmentation tree for q (like Fig. 1). A tree ensures a unique shortest path between t_1 and t_2, which is through the common ancestor of t_1 and t_2. For example, $td($`xp`, `video`; $n(q)$ in Fig. 1$) = 7$. The minimum possible tree distance between two terms is two. Note that td between t_1 and t_2 can vary for the same q, depending on $n(q)$.

Query distance. The query distance $qd(t_1, t_2; q)$ between two terms t_1 and t_2 in a query q is defined as the difference between the positions of t_1 and t_2 in q, or equivalently, the number of intervening words plus one.

Document distance. Let t_1 and t_2 be two terms in the query q, which are also present (matched) in a retrieved document $\mathcal{D}$. Let there be k instances of *ordered* pairwise occurrences of t_1 and t_2 (*ordered* pairs of positions of t_1 and t_2, (p_1, p_2) where $p_1 < p_2$) in $\mathcal{D}$ at minimum distances [6] $dist_i = dist_1, dist_2, \ldots, dist_k$, such that the $dist_i$-s are in ascending order. We combine the ideas of minimum distance and multiple occurrences of a term pair to formulate the following definition of accumulative inverse document distance ($AIDD$) for t_1 and t_2 in $\mathcal{D}$:

$$AIDD(t_1, t_2; \mathcal{D})_{t_1 \neq t_2} = \frac{1}{dist_1} + \frac{1}{dist_2} + \ldots + \frac{1}{dist_k} \tag{1}$$

By this method, a document with several (t_1, t_2) near to each other will have a high $AIDD$. Since our concept is based on minimum distance, we do not need a document length normalizer. A threshold on k is nevertheless necessary to avoid considering *all* pairwise distances of t_1 and t_2, as distant pairs could be semantically unrelated. To avoid scoring unrelated occurrences of a term pair, we consider matches only if (t_1, t_2) occur within a given window size, win.

[5] For all distances, when the same word appears multiple times in a query, each word instance is treated as distinct during pairwise comparisons.

2 Algorithm and IR application

2.1 Splitting and joining flat segments

Since flat segmentation is a well-researched problem, we develop our algorithm for nested segmentation by starting with a flat segmentation of the query and trying to *split* within a flat segment and *join* adjacent flat segments recursively. Our main motivation for designing simple segment nesting strategies stems from the fact that most flat segmentation algorithms compute scores for n-grams as a key step of their respective methods (generally $n \leq 5$) [3–5]. In doing so, most often, the scores of the contiguous lower order n-grams $(n-1, n-2, \ldots)$ are also known. For **splitting a flat segment**, we exploit these scores to deduce the structure within a flat segment. In this work, we specifically use the state-of-the-art Co-occurrence Significance Ratio (CSR) measure [7] to score n-grams. We adopt a simple greedy approach in this research. The n-gram (here $n \leq 3$) that has the highest CSR score within a flat segment (where the number of words in the n-gram is less than the number of words in the corresponding flat segment) is immediately grouped together as a unit, i.e. a *sub-segment.* We define a *sub-segment* as a smaller segment created by the division of a larger segment. Recursively, this newly grouped sub-segment's left and right n-grams (possibly null) and the sub-segment itself are processed in the same greedy fashion till every string to be processed cannot be divided further.

Joining flat segments is essential to completing the nested segmentation tree, which in turn ensures a path between every pair of words in the query. The bigram at a flat segment boundary, i.e. the last word of a flat segment and the first word of the next flat segment, can be effectively used to take the segment joining decision. In our running example, if we wish to decide whether to join `windows xp home edition` and `hd video`, *or* `hd video` and `playback`, we check the relative order of the scores of the (ordered) bigrams formed by the underlined words only. The bigram with the higher score (in this case `video playback`) dictates which pair should be joined. This process is similarly repeated on the new parenthesized segments obtained until the whole query forms one unit. Here we use pointwise mutual information (PMI) [5] to score bigrams. It often happens that the last (or the first) word in a segment is a determiner, conjunction or preposition (DCP) (list used from `http://goo.gl/Ro1eeA`). In these cases, it is almost always meaningful to combine such a segment with the next segment (or the previous segment) to make a meaningful *super-segment* (a larger segment created by the joining of two smaller segments). Examples are `(bed and) (breakfast)` and `(sound) (of music)`. We prioritize such cases over the bigram scores during the joining process.

2.2 Using nested segmentation in IR

We define a score *Re-rank Status Value* ($RrSV$) of every document $\mathcal{D}$ that was retrieved in response to an unsegmented query q, determined by the following principle – *a pair of words that has a low tree distance in the nested representation* *of the query should not have a high document distance.* In other words, while re-ranking a document, the document distance (Eq. 1) between a pair of words should be penalized by a factor *inversely* proportional to its tree distance. The $RrSV$ for a document $\mathcal{D}$ is thus defined as

$$RrSV_{\mathcal{D}} = \sum_{\substack{t_i, t_j \in q \cap \mathcal{D} \\ t_i \neq t_j \\ td(t_i, t_j; n(q)) < \delta}} \frac{AIDD(t_i, t_j; \mathcal{D})}{td(t_i, t_j; n(q))} \tag{2}$$

where t_i-s are query terms matched in the document and $n(q)$ is the nested segmentation for q. However, we do not wish to penalize $\mathcal{D}$ when the words are close by in the document and are relatively far in the tree. This analysis drives us to create a tree distance threshold (cut-off) parameter δ. In other words, if $td(a, b; n(q)) < \delta$, only then is the word pair a and b considered in the computation of $RrSV$. The original rank for a page (obtained using TF-IDF scoring, say) and the new rank obtained using $RrSV$ are fused using the method in Agichtein et al. [8], using w as a tuning parameter. We refer to this entire strategy as the *Tree* model. We use three re-ranking baselines: *Flat segmentation* (word pairs are limited to cases where both words come from a single flat segment), document distances only (no scaling using tree distance; *Doc* model), and query distances (scaling document distances using query distances (Sec. 1); *Query* model).

3 Datasets and experimental results

3.1 Datasets

Our nested segmentation algorithm requires a query log as the only resource, for computing the various n-gram scores. For our experiments, we use a query log sampled from a Bing Australia in May 2010. This raw data slice consists of $16.7M$ queries ($4.7M$ unique). In order to ensure the replicability of our results, we report our IR evaluation on publicly available datasets only and use open source retrieval systems. We used the dataset released by Saha Roy et al. [2], and refer to it as SGCL12 (author last name initials and year), using Apache Lucene v.3.4.0 as the search engine. The first 250 queries were used as the development set for tuning model parameters (k, win, δ and w) and the last 250 queries were used as the test set. We also used a collection of 75 TREC queries sampled from the Web tracks of 2009 to 2012, with ≥ 3 words and at least one relevant document in the top-100 results. The Indri search engine was used along with the ClueWeb09 dataset. 35 queries were used as the development set for tuning model parameters and the remaining 40 queries were used as the test set, and the results are averaged over ten random 35-40 splits.

3.2 Experiments and results

We used the outputs of three recent flat segmentation algorithms as input to the nested segmentation algorithm and final nested segmentations for these

Table 1. Comparison of flat and nested segmentations on SGCL12 and TREC-WT.

Dataset	**Algo**	**Hagen et al. [5]**		**Mishra et al. [4]**		**Saha Roy et al. [2]**		**Huang et al. [9]**
SGCL12	**Unseg**	**Flat**	**Nested**	**Flat**	**Nested**	**Flat**	**Nested**	**Nested**
nDCG@5	0.6839	0.6815	**0.6982**	**0.6977**	0.6976	0.6746	**0.7000**†	0.6996
nDCG@10	0.6997	0.7081	**0.7262**†	0.7189	**0.7274**	0.7044	**0.7268**†	0.7224
nDCG@20	0.7226	0.7327	**0.7433**†	0.7389	**0.7435**	0.7321	**0.7433**†	0.7438
MAP	0.8337	0.8406	**0.8468**†	0.8411	**0.8481**†	0.8423	**0.8477**	0.8456
TREC-WT	**Unseg**	**Flat**	**Nested**	**Flat**	**Nested**	**Flat**	**Nested**	**Nested**
nDCG@5	0.1426	0.1607	**0.1750**†	0.1604	**0.1752**†	0.1603	**0.1767**†	0.1746
nDCG@10	0.1376	0.1710	**0.1880**†	0.1726	**0.1882**†	0.1707	**0.1884**†	0.1845
nDCG@20	0.1534	0.1853	**0.1994**†	0.1865	**0.2000**†	0.1889	**0.2010**†	0.1961
MAP	0.2832	0.2877	**0.3298**†	0.3003	**0.3284**†	0.3007	**0.3296**†	0.3263

Statistical significance of nested segmentation (under the one-tailed paired t-test, $p < 0.05$) over flat segmentation *and* the unsegmented query is marked using †.

queries were obtained. Documents are retrieved using unsegmented queries, and re-ranked using the proposed technique and the baselines. Results are compared in terms of nDCG@k ($k = 5, 10, 20$; the IDCG is computed using the optimal ranking from all judgments for a query) and MAP (URLs with ratings > 0 were considered as relevant). For each dataset, the four parameters k, win, δ and w were optimized using the grid search technique for maximizing nDCG@10 on the development set. Partial data and complete code for this project is available at `http://cse.iitkgp.ac.in/resgrp/cnerg/qa/nestedsegmentation.html`.

(a) Improvements over flat segmentation: Some of the sample outputs from our nested segmentation algorithm are as follows: `((garden city) (shopping centre)) (brisbane qld)`, `(the ((chronicles of) riddick)) (dark athena)`, and `((sega superstars) tennis) ((nintendo ds) game)`. In Table 1, for each algorithm, *Flat* refers to the re-ranking strategy for flat segmentation by the corresponding algorithm, and *Nested* refers to the tree re-ranking strategy when applied to the nested segmentation generated when the corresponding flat segmentation was used as the start state. We observe that nested segmentation significantly outperforms the state-of-the-art flat segmentation algorithms in all cases. Importantly, improvements are observed for both datasets on all metrics. This indicates that one should not consider proximity measures for *only* pairs of terms that are within a flat segment. Thus, our experiments provide evidence against the hypothesis that a query is similar to a bag-of-segments [4]. We also note that both the flat and nested segmentations perform better than the unsegmented query, highlighting the general importance of query segmentation for IR.

(b) Comparison with past work: We apply our re-ranking framework on the nesting output by Huang et al. [9] and show results in Table 1. We observed that their method is outperformed by the proposed nested segmentation (from all input flat segmentation strategies) on several metrics. We observed that while the average tree height is 2.96 for our nesting strategy, the same is about 2.23 for Huang et al. (SGCL12). Note that due to the strict binary partitioning at each step for Huang et al., one would normally expect a greater average tree height for this method. Thus, it is the inability of Huang et al. to produce a suitably deep tree for most queries (inability to discover fine-grained concepts) that is

responsible for its somewhat lower performance. Most importantly, all nesting strategies faring favorably (none of the differences for Huang et al. with other nesting methods are statistically significant) with respect to flat segmentation bodes well for the usefulness of nested segmentation.
(c) Comparison of re-ranking strategies: We find the *Tree* model performs better than *Doc* and *Query* models. We observed that the number of queries on which *Doc*, *Query* and *Tree* perform the best (possibly multiple strategies) are 102, 94, 107 (SGCL12, 250 test queries) and 30, 29.7, 30.8 (TREC-WT, 40 test queries, mean over 10 splits) respectively.

4 Conclusions

This research is one of the first systematic explorations of nested query segmentation. We have shown that the tree structure inherent in the hierarchical segmentation can be used for effective re-ranking of result pages ($\simeq 7\%$ nDCG@10 improvement over unsegmented query for SGCL12 and $\simeq 40\%$ for TREC-WT). Importantly, since n-gram scores can be computed offline, our algorithms have minimal runtime overhead. We believe that this work will generate sufficient interest and several improvements over the present scheme would be proposed in the recent future. In fact, nested query segmentation can be viewed as the first step towards query parsing, and can lead to a generalized query grammar.

Acknowledgments

The first author was supported by Microsoft Corporation and Microsoft Research India under the Microsoft Research India PhD Fellowship Award.

References

1. Li, Y., Hsu, B.J.P., Zhai, C., Wang, K.: Unsupervised query segmentation using clickthrough for information retrieval. In: SIGIR '11. (2011) 285–294
2. Saha Roy, R., Ganguly, N., Choudhury, M., Laxman, S.: An IR-based evaluation framework for web search query segmentation. In: SIGIR '12. (2012) 881–890
3. Tan, B., Peng, F.: Unsupervised query segmentation using generative language models and Wikipedia. In: WWW '08. (2008) 347–356
4. Mishra, N., Saha Roy, R., Ganguly, N., Laxman, S., Choudhury, M.: Unsupervised query segmentation using only query logs. In: WWW '11. (2011) 91–92
5. Hagen, M., Potthast, M., Stein, B., Bräutigam, C.: Query segmentation revisited. In: WWW '11. (2011) 97–106
6. Cummins, R., O'Riordan, C.: Learning in a pairwise term-term proximity framework for information retrieval. In: SIGIR '09. (2009) 251–258
7. Chaudhari, D.L., Damani, O.P., Laxman, S.: Lexical co-occurrence, statistical significance, and word association. In: EMNLP '11. (2011) 1058–1068
8. Agichtein, E., Brill, E., Dumais, S.: Improving web search ranking by incorporating user behavior information. In: SIGIR '06. (2006) 19–26
9. Huang, J., Gao, J., Miao, J., Li, X., Wang, K., Behr, F., Giles, C.L.: Exploring web scale language models for search query processing. In: WWW '10. (2010) 451–460