Two-Stage Query Segmentation for Information Retrieval

Michael Bendersky
bemike@cs.umass.edu

W. Bruce Croft
croft@cs.umass.edu

David Smith
dasmith@cs.umass.edu

Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

ABSTRACT

Modeling term dependence has been shown to have a significant positive impact on retrieval. Current models, however, use sequential term dependencies, leading to an increased query latency, especially for long queries. In this paper, we examine two query segmentation models that reduce the number of dependencies. We find that two-stage segmentation based on both query syntactic structure and external information sources such as query logs, attains retrieval performance comparable to the sequential dependence model, while achieving a 50% reduction in query latency.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Query Formulation

General Terms: Algorithms, Experimentation, Theory

Keywords: Query segmentation, term dependence, long queries

1. INTRODUCTION

Modeling term dependencies can have a significant positive impact on retrieval performance [1, 3, 4, 6, 7]. For instance, Metzler and Croft [6] show that a *sequential dependence model*, which uses all ordered adjacent pairs of query terms, is highly effective for retrieval on TREC corpora. Bai et al. [1] demonstrate similar results on a web search engine.

In the web search setting, where query latency is crucial, the computation of all sequential dependencies may become prohibitively expensive for long queries, as the number of dependencies grows linearly with the size of the query. Hence, it would be beneficial to reduce the number of dependencies, while still retaining the improvements in retrieval effectiveness brought on by modeling term dependencies.

Recent work on *query segmentation* [2, 3, 4, 5, 8] seeks to address this problem. For natural language queries (e.g., TREC topic descriptions), a segmentation can be obtained by a shallow syntactic parser [4]. For non-grammatical long queries, often encountered in web search (e.g., *"new york times square"*), supervised segmentation that leverages information from external sources such as query logs, web corpora and Wikipedia has been shown to be more effective [2, 5, 8]. In fact, the segmentation techniques need not be mutually exclusive. As we show, a supervised segmenter trained on an external data source can be applied as a second stage for a better segmentation of long noun phrases.

SIGIR'09, July 19–23, 2009, Boston, Massachusetts, USA.
ACM 978-1-60558-483-6/09/07.

Model	Query Segmentation	AP
IDM	members rock group nirvana	26.84
SDM	[members rock] [rock group] [group nirvana]	27.44
NDM	members [rock group nirvana]	26.63
TDM	members [rock group] nirvana	29.20

Table 1: Representation of the query *"Find information on members of the rock group Nirvana"* under the four segmentation models (with stopwords removed). Average precision under each model is also reported.

There is little work on investigating the effectiveness of query segmentation for retrieval. Recent work by Guo et al. [5] shows that such segmentation (among other factors) contributes to the effectiveness of web search. However, there is no direct comparison, to the best of our knowledge, between using supervised query segmentation and sequential dependence for retrieval with long queries.

2. MODEL

The model described here has two orthogonal dimensions: (i) retrieval model, and (ii) segmentation model. The focus of this paper is on the latter, and hence we use the existing MRF model [6] for the former. MRF scores a document by:

$$score(d) \triangleq \sum_{t \in T} \lambda_T f_T(t) + \sum_{s \in S} \lambda_O f_O(s) + \sum_{s \in S} \lambda_U f_U(s), \quad (1)$$

where T is the set of all query terms, and S is the set of all query segments. Functions f_T, f_O, f_U calculate the term matches, exact segment matches and unordered segment matches, respectively. λ's are the model parameters[1].

The existing and the new query segmentation models we consider in this paper vary by their independence assumptions. Table 1 gives an example of a query representation under each of these segmentation models.

As benchmarks, we consider two well known segmentation models. The independence model, `IDM`, sets $S = \emptyset$, and is, in fact, equivalent to the unigram language model [6]. The sequential dependence model [6], `SDM`, assumes dependence between all adjacent term pairs. While being more effective than `IDM`, `SDM` is also much more expensive to compute, especially for long queries.

To ammeliorate the computational overhead incurred by `SDM`, we propose two novel segmentation models. The noun phrase dependence model, `NDM`, assumes that all the terms

[1]We set λ's to $[0.85, 0.1, 0.05]$, as this setting was shown to be nearly-optimal for the sequential dependence model [6].

Feature	Context	Description	Source
google_gram(n)	$n = 1, \ldots, 4$	n-gram count in a large web collection	LDC Catalog # LDC2006T13
wiki_intitle(n)	$n = 1, \ldots, 4$	n-gram count of partial matches with Wikipedia titles	`http://download.wikimedia.org/enwiki/`
wiki_exact_title(n)	$n = 1, \ldots, 4$	n-gram count of exact matches with Wikipedia titles	`http://download.wikimedia.org/enwiki/`
msn_inquery(n)	$n = 1, \ldots, 4$	n-gram count of partial matches with web queries	Microsoft 2006 RFP dataset
msn_exact_query(n)	$n = 1, \ldots, 4$	n-gram count of exact matches with web queries	Microsoft 2006 RFP dataset
qry_pos	$n = 1, 2$	Position of a token in a query (forward, backward)	
POS_tag	$n = 1, 2$	Part-of-speech tag of the token	

Table 2: Features for the second stage of query segmentation. n is the context window size considered at each token.

	ROBUST04 (Topics 301-450, 601-700)				*W10g* (Topics 451-550)				*GOV2* (Topics 701-850)			
	cpu-t	P@5	b-pref	MAP	*cpu-t*	P@5	b-pref	MAP	*cpu-t*	P@5	b-pref	MAP
`IDM`	*188*	49.32	25.54	24.88	*116*	40.40	19.72	18.78	*2,390*	52.62	34.24	24.85
`SDM`	*781*	**50.44**	**26.30***	**25.84***	*500*	41.00	20.73	19.81*	*16,488*	**56.78***	35.85*	**27.11***
`NDM`	*438*	50.12	25.49†	24.78†	*278*	41.00	20.89	19.72	*8,115*	54.50	36.14*	26.60*
`TDM`	*450*	50.28	25.89*	25.16†	*277*	**42.00**	**21.09***	**20.11***	*7,785*	55.84*	**36.18***	26.85*

Table 3: Retrieval efficiency and effectiveness. Statistically significant differences (Wilcoxon sign test, $\alpha < 0.05$) with *IDM* and *SDM* are indicated by $*$ and †, respectively. Best result in column is marked in bold.

within the boundaries of a noun phrase chunk are dependent, and there are no dependencies between the chunks. The two-stage segmentation model, `TDM`, further segments the long[2] noun phrase chunks into finer-grained concepts. Second stage segmentation of the noun phrase chunks uses not only POS tags, but also features that were found to be useful for query segmentation [2, 8] (see Table 2).

3. EVALUATION

In this section, we evaluate the retrieval performance attained by the proposed segmentation models. In our experiments, we use three TREC corpora: one newswire collection (ROBUST04) and two web collections (W10g and GOV2). Indexing and retrieval is done using the Indri[3] toolkit. The indexes are stemmed using a Porter stemmer. For computing functions f_O, f_U in Eq. 1, we use Indri ordered window (`#OW1`) and unordered window (`#UW8`) operators, respectively.

The queries are stopped using a custom list of 52 stopwords, designed to remove some of the frequent stop patterns in *description* queries (e.g., "find information"). In all retrieval experiments, Dirichlet smoothing with $\mu = 1500$ is used. Only the *description* portion of TREC topics is used.

Noun phrase chunks are extracted using a syntactic shallow parser CRFChunker[4] trained on WSJ corpus. Second stage segmentation is done using a sequential multi-purpose chunker YamCha[5]. For training the second stage segmentation model we use a corpus of 500 pre-segmented noun phrases [2]. Table 2 details the features used in training.

Table 3 summarizes the retrieval efficiency and effectiveness of the proposed query segmentation models. For efficiency, we report the CPU time required to run all the queries. For effectiveness, we report P@5, MAP and b-pref.

In terms of effectiveness, all segmentation models outperform the independence model. `SDM` and `TDM` are the best performing methods and have no statistically significant differences, save for MAP on ROBUST04. `TDM` always outperforms `NDM`, verifying our assumption about the utility of applying the finer-grained second segmentation stage.

[2]Second stage segmentation is applied to noun phrases that contain more than two terms.

[3]http://www.lemurproject.org/indri/

[4]http://crfchunker.sourceforge.net/

[5]http://chasen.org/~taku/software/YamCha/

In terms of efficiency, `SDM` is significantly slower than the other models. Both `NDM` and `TDM` decrease the CPU time by 50% compared to `SDM`, while retaining most of its effectiveness gains, or even improving effectiveness (`TDM` on W10g). Note that in an operational system, n-gram indexing could further reduce query latency for all the dependence models.

4. CONCLUSIONS

In this paper, we have shown that query latency can be greatly reduced, without compromising effectiveness, by a two-stage segmentation process that takes into account both query syntax and external information sources such as query logs. In future work, we intend to further investigate segmentation models that can efficiently and effectively incorporate non-adjacent terms and concept weighting.

5. ACKNOWLEDGMENT

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #IIS-0534383 and in part by Yahoo! Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

6. REFERENCES

[1] J. Bai, Y. Chang, H. Cui, Z. Zheng, G. Sun, and X. Li. Investigation of partial query proximity in web search. In *Proc. of WWW*, pages 1183–1184, 2008.

[2] S. Bergsma and Q. I. Wang. Learning noun phrase query segmentation. In *Proc. of EMNLP-CoNLL*, pages 819–826. Association for Computational Linguistics, 2007.

[3] J. Gao, J.-Y. Nie, G. Wu, and G. Cao. Dependence language model for information retrieval. In *Proc. of SIGIR*, pages 170–177, 2004.

[4] J. Gao, H. Qi, X. Xia, and J.-Y. Nie. Linear discriminant model for information retrieval. In *Proc. of SIGIR*, pages 290–297, 2005.

[5] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *Proc. of SIGIR*, pages 379–386, 2008.

[6] D. Metzler and W. B. Croft. A Markov random field model for term dependencies. In *Proc. of SIGIR*, pages 472–479, 2005.

[7] G. Mishne and M. de Rijke. Boosting web retrieval through query operations. In *Proc. of ECIR*, pages 502–516, 2005.

[8] B. Tan and F. Peng. Unsupervised query segmentation using generative language models and Wikipedia. In *Proc. of WWW*, pages 347–356, 2008.