On the Fly Query Segmentation Using Snippets

David J. Brenes[1], Daniel Gayo-Avello[2] and Rodrigo Garcia[3]

[1] Simplelogica S.L. david.brenes@simplelogica.net
[2] University of Oviedo dani@uniovi.es
[3] University of Oviedo rodrigo@innova.uniovi.es

Abstract. One of the most important issues in Information Retrieval is inferring the intents underlying users' queries, and query segmentantion, provided it is done fast, can be a valuable tool. Such techniques usually rely on a prior training phase involving large datasets. A costly process, specially in environments which are increasingly moving towards real time scenarios where latency to retrieve fresh informacion should be minimal. In this paper an 'on-the-fly' query segmentation method is proposed. It uses snippets which are later mined by means of a naive statistical algorithm. An initial evaluation of such a method is provided, in addition to a discussion on its applicability to different scenarios.

1 Introduction

One of the main purposes of Web Information Retrieval is helping users to fulfill their information needs. However, most of the time, the main problem is determining the users' intent. This is a complex task that can be improved by using a number of different techniques.

One useful approach is extracting MWUs[4] from the queries sent by the user. Unfortunately, query segmentation tend to be performed by means of statistical or machine learning methods which rely on a prior large corpus of training data which is processed in advance during the training phase because of the unfeasibility of doing it in real time when the queries are submitted.

Hence, performance is achieved at the cost of being unable to react to new types of queries. This can affect a considerable amount of users because a significant number of queries are submitted just once to the information retrieval system ([6], [1], [7]). More importantly, this issue is critical in an ecosystem driving towards 'real time' where users demand fresh content faster than ever.

In this paper the authors propose and evaluate a new query segmentation approach designed to be as light and fast as possible; avoiding costly preprocessing or large datasets. The main goal is being able to create a system which can adapt to real time changes by just analyzing the pertinent information for the current query.

[4] Multi-Word Units: a relevant group of words

2 Previous Research

Nonetheless to say, noun phrase segmentation is not a novel area and large research efforts have been applied to this question ([5], [8], [3] or [2]).

Concerning Web Information Retrieval, noun phrase segmentation has been applied to query segmentation in the same way it is applied to longer documents; that is, without taking into account the special features and requirements of search queries: shorter, much more schematic, with a very different vocabulary set.

3 Proposal Description

3.1 Characteristics Overview

A new Web enviroment is moving towards an ecosystem of real time applications where the context changes very quickly as the news spread. This leads to the main requirement of our approach: to offer good results not only for unexpected queries, but also for queries related to drifting topics of interest.

We are not saying, by no means, that methods relying on a prior training are not fast or feasible for this task. They are extremely fast (once trained) and they have been applied to query segmentation (e.g. [8]). It is the speed of topic drifting and how to handle that changing trends which we feel can't be properly achieved by using methods relying on training data and heavy result caching. Because of this, our approach consists of performing query segmentation on-the-fly. Of course, a balance should be reached as the results must be provided in a reasonable amount of time and, in fact, repeated queries could and should be cached in order to avoid duplicate computations.

Authors are also conscious that returning fresh results demands a light algorithm which may not achieve as good performance as previous approaches based on training data. However, we feel that a certain lack of precision could be tolerable in exchange for greater adaptability to drifting and evolving topics.

3.2 Snippet Sources

Our method exploits snippets which are short extracts of web documents usually containing one or more keywords from the user query. They are commonly seen in result pages of search engines, but we are open to also consider other sources. However, for the purpose of this paper, we have only used snippets from three major search engines (Google[5], Bing[6] and Yahoo![7]) exploiting their respective APIs. This way, we have been able to compare the performance of our method using different sources of snippets.

[5] http://code.google.com/intl/es-ES/apis/ajaxsearch/documentation/#fonje
[6] http://msdn.microsoft.com/en-us/library/dd251056.aspx
[7] http://developer.yahoo.com/search/boss/

3.3 Statistics Measures

In addition to our own method, we tested several statistical measures previously employed in MWU detection; namely, Mutual Information, SCP, Phi, Dice and LogLike. [4] devised a way to generalize those well-known measures from bigrams to n-grams of arbitrary length and, hence, we have used those generalized versions.

As we have already said, we have devised another method based on the use of snippets. The underlying idea is really simple: First, all possible n-grams in the query are produced. Then, the frequency for each n-gram in the snippets is to be found and those n-grams below a certain threshold are discarded. Finally, the query is segmented by using the most frequent remaining n-grams. The aforementioned threshold limits the minimum number of snippets a n-gram should appear in (for instance, a 0.5 threshold requires the n-gram to appear in half or more of the snippets).

We tested four different values for this parameter (namely, 0.25, 0.5, 0.75 and 1) so we can study how it affects to the performance of our method. However, because of space costraints only results for 0.25 and 1 values are shown.

4 Evaluation

4.1 Datasets

In order to evaluate the different query segmentation methods we needed segmented queries. Such queries could have been segmented by experts (e.g. the authors themselves or other scholars) or, much better, by users of a search engine. In the first case, one could assume a consistent set of segmentations while in the second one the data could be noisier but also larger and closer to the actual use scenario.

We were able to use one dataset of each kind. For the expert-segmented data we relied on a subset of the AOL query log [9] manually segmented by [3]; we will refer to this dataset as aol-bergsma-wang-2007 from now on. The user-segmented data was collected by the authors of this study using the SearchSpy feature from the Dogpile metasearch engine. This second dataset (searchspy-2010 from now on) just comprises queries with paired double quotes enclosing two or more terms.

4.2 Evaluation Methodology

At a first glance, one could think of using the percentage of correctly segmented queries to measure the effectiveness of the different methods. However, this approach does not seem to be the best choice because (1) we can think of different degrees of segmentation accuracy (`[new york] [travel guides]` better than `[new york] travel guides` better than `new [york travel] guides`) and (2) segmentation is a user-dependent task and two different users –even experts, as shown by [3]– can provide different segmentations.

Thus, we chose to use the well-known measures of precision and recall as used in other segmentation tasks; that is, taking into account not the queries but the blanks between segments both in the reference and the segmentation provided by each algorithm. For instance, in the query `new york travel guides` and assuming the correct segmentation is `[new york] [travel guides]`, the segmentation `[new york] travel guides` have a precision of 1 and a recall of 0.5, while segmentation `new [york travel] guides` have both 0 precision and 0 recall.

4.3 Positive Bias from Snippets Sources

We run the algorithms in three different 'flavors'. Each flavor rearranged the queries in a different way before submitting it to the search engine to obtain the snippets: (1) removing all the double-quotes; (2) sending the original queries; and (3) removing the double-quotes and reversing the order of the terms.

This way, we have (1) a base case in which queries are submitted with no segmentation information but keeping keyword ordering; (2) a best case where all the segmentations within the query are preserved; and (3) a worst case where neither segmentations nor keyword ordering are preserved.

By doing this it could be possible to appreciate, albeit somewhat indirectly, the impact term collocation could exert on the results and, consequently, on the snippets and the different methods performance. Thus, if term collocations were heavily used by search engines to produce results then the differences between flavors 1 and 2 should be minimal. Additionally, by reversing the order of the terms in the query all valid collocations are removed and spurious ones are introduced; this should make much harder to find valid segmentations. If even under this hard circumstances the methods manage to find correct segmentations we could safely discard the hypothesis of the search engines performing query segmentation and, thus, we could argue that the proposed methods are actually performing query segmentation by just relying on short text snippets.

5 Experiment Results

5.1 Performance on *searchspy-10* Query Log

Table 1 shows the P, R, and F measures for each statistic measure and each snippet source used in the experiments.

When Yahoo! Boss or Bing are used as snippet sources, F measures vary between 0.6 to 0.77; this is a reasonable performance for a technique that is using relatively little information (10 snippets of text as a maximum) which would make statistical algorithms to work in an unreliable way.

Quite surprisingly, using Google as the snippet source drops F measures to very low values (from 0.48 to 0.61). The main reason for such differences, beyond the quality of the returned results, is the amount of queries from the query log that Google couldn't find snippets for. About 1,100 against the less

Measure	Snippet Source	P	R	F
Mutual Information	Bing	0.6834	0.5481	0.6158
	Boss	0.7464	0.6210	0.6837
	Google	0.5374	0.4349	0.4862
SCP	Bing	0.6327	0.6216	0.6271
	Boss	0.6609	0.6481	0.6545
	Google	0.6145	0.6089	0.6117
Phi	Bing	0.6888	0.5367	0.6128
	Boss	0.7530	0.6061	0.6795
	Google	0.5411	0.4246	0.4829
Dice	Bing	0.6888	0.5354	0.6121
	Boss	0.7519	0.6057	0.6788
	Google	0.5405	0.4241	0.4823
Loglike	Bing	0.7053	0.6383	0.6718
	Boss	0.7372	0.6663	0.7017
	Google	0.5349	0.4715	0.5032
Entity Frequency (25)	Bing	0.7336	0.6446	0.6891
	Boss	0.8089	**0.7375**	**0.7732**
	Google	0.5873	0.5298	0.5585
Entity Frequency (100)	Bing	0.7530	0.5095	0.6312
	Boss	**0.8329**	0.5701	0.7015
	Google	0.6030	0.4084	0.5057
Average	Bing	0.7090	0.5753	0.6422
	Boss	**0.7717**	**0.6401**	**0.7059**
	Google	0.5734	0.4712	0.5223

Table 1. Performance using different statistic measures and snippet sources for queries in *searchspy-10* query log

than 500 for Bing and less than 300 for Yahoo!. However, the proposed system is higly dependant on the snippet source so we feel it was important to measure performance using the whole query log, 'failed' queries included.

Concerning the statistical methods, *Entity Frequency* –the simplest one– consistently achieves the best performance (according to F-measure); outperforming *Loglike*, the second best algorithm, by 10.19%.

5.2 Performance on *aol-bergsma-wang-2007* Query Log

In table 2 we can see the results of our method when running on *aol-bergsma-wang-2007* query log.

For this query log, F measures achieves better performance (between 0.07 for Boss and 0.23 for Google). The main reason for this better performance is that very few queries obtained no results in the snippet sources because [3] selected only those queries with at least one clicked result in the AOL query log.

Measure	Snippet Source	P	R	F
Mutual Information	Bing	0.7666	0.7240	0.7453
	Boss	0.7974	0.7557	0.7765
	Google	0.7833	0.7335	0.7584
SCP	Bing	0.7084	0.7507	0.7295
	Boss	0.7234	0.7752	0.7493
	Google	0.6953	0.7512	0.7233
Phi	Bing	0.8159	0.7362	0.7760
	Boss	0.8421	0.7655	0.8038
	Google	0.8269	0.7527	0.7898
Dice	Bing	0.8182	0.7423	0.7802
	Boss	0.8387	0.7644	0.8016
	Google	0.8256	0.7545	0.7901
Loglike	Bing	0.7842	0.8038	0.7940
	Boss	0.8012	**0.8264**	**0.8138**
	Google	0.8053	0.8238	0.8145
Entity Frequency (25)	Bing	0.7183	0.7895	0.7539
	Boss	0.7374	0.8162	0.7768
	Google	0.7150	0.7886	0.7518
Entity Frequency (100)	Bing	0.8427	0.5430	0.6929
	Boss	**0.8733**	0.5590	0.7161
	Google	0.8505	0.5534	0.7019
Average	Bing	0.7848	0.7149	0.7499
	Boss	**0.8069**	**0.7396**	**0.7732**
	Google	0.7901	0.7259	0.7580

Table 2. Performance using different statistic measures and snippet sources for queries in *aol-bergsma-wang-2007* query log

If those queries without results are removed the differences between snippet sources are not significant, but, again, using Yahoo! Boss achieves the better performance.

Concerning the algorihms, *Entity Frequency* behaves better with this query log than with the previous one but a more important boost is the one achieved by other statistic measures such as *Dice*, *Mutual Information* or *Loglike*. In fact, *Loglike* is the algorithm which achieves the highest values for R and F measures, although *Entity Frequency* still shows the highest value for Precision.

5.3 Bias Results

Table 3 shows the variation obtained in F measure for the best and worst cases as described in Section 4.3.

To compute the data in that table, queries with no results were removed because they were different for each search engine and, thus, they introduced an additional source of bias. By removing them, we can compare search engines

to each other and, thus, have a glimpse on the way they can (or cannot) be performing query segmentation by themselves.

Source	*searchspy-10*			*aol-bergsma-wang-2007*		
	Original	Quoted	Reversed	Original	Quoted	Reversed
Bing	0.799	0.889 (+11%)	0.796 (-0.37%)	0.771	0.831 (+7.81%)	0.779 (+1.04%)
Boss	0.808	0.872 (+7.98%)	0.792 (-1.98%)	0.787	0.840 (+6.71%)	0.783 (-0.5%)
Google	0.805	0.879 (+9.22%)	0.793 (-1.54%)	0.780	0.850 (+8.98%)	0.780 (-0.06%)

Table 3. Average F measures for best (quoted), base (original) and worst (reversed) scenarios

When we compare best case diferences against worst cases ones, we find that best cases are very different from the base case. So we can safely assert that (1) our technique is actually performing query segmentation –certainly by means of the snippets– and (2) it is not inadvertently taking advantage of any underlying query segmentation or entity detection phase performed by the search engine.

6 Conclussions

6.1 Summary

Query segmentation can be a useful tool to tackle with the currently emerging real-time scenario for IR systens. Accordingly, we have proposed the use of short snippets to obtain –by means of simple statistical methods– the most relevant MWUs. In addition to that, we performed several experiments on two previously segmented query logs: one by experts and the other by actual Web searchers. Preliminary results are really promising (F-measure is about 0.8) and, thus, further research in this line should be done.

6.2 Snippet Source Dependency

searchspy-10 query log reflects real users' behavior with regards to query segmentation. The main conclusion one can reach from the experiments conducted on this query log is that performance is highly dependent on the underlying snippet source. For example, preliminary work using Wikipedia search engine has shown it can be a very good source of snippets but only for certain kind of queries.

6.3 Statistic algorithm election

Two algorithms outperform the others in both experiments: Loglike and Entity Frequency: Loglike achieves the highest performance for all the experiments,

but its behavior is highly irregular and does not handle very well queries from 'actual' users. Entity Frequency provides much more consistent results for both query logs achieving similar performance. Besides, Entity Frequency different performance values between thresholds is also a variable to take into account.

7 Future Work

We have described a preliminary implementation which achieves good results; nevertheless, some questions remain open and they deserve further research.

- *Real Time Snippets*:
  More work must be done on integrating different snippet sources directly related to real-time Web.
- *Snippet Sources Integration*:
  The use of various complementary sources of snippets could improve the performance.

8 Acknowledgments

This work was partially financed by grant UNOV-09-RENOV-MB-2 from the University of Oviedo

References

[1] Steven M. Beitzel, Eric C. Jensen, David D. Lewis, Abdur Chowdhury, and Ophir Frieder. Automatic classification of Web queries using very large unlabeled query logs. *ACM Transactions on Information Systems (TOIS)*, 25(2), 2007.

[2] Michael Bendersky, W. Bruce Croft, and David a. Smith. Two-stage query segmentation for information retrieval. *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval - SIGIR '09*, page 810, 2009.

[3] Shane Bergsma and Qin Iris Wang. Learning Noun Phrase Query Segmentation. *Computational Linguistics*, (June):819–826, 2007.

[4] Joaquim Ferreira Da Silva and Gabriel Pereira Lopes. A Local Maxima method and a Fair Dispersion Normalization for extracting multi-word units from corpora. *Sixth Meetings on Mathematics of Language*, pages 369–381, 1999.

[5] Jianfeng Gao, Jian-Yun Nie, Guangyuan Wu, and Guihong Cao. Dependence language model for information retrieval. *Annual ACM Conference on Research and Development in Information Retrieval*, 2004.

[6] Bernard J. Jansen, Amanda Spink, Tefko Saracevic, and Judy Bateman. Real life information retrieval: a study of user queries on the Web. *SIGIR Forum*, 32(1):5–17, 1998.

[7] Bernard. J. Jansen, Amanda Spink, Tefko Saracevic, and Dietmar Wolfram. Searching the Web: The Public and Their Queries. *Journal of the American Society for Information Science and Technology*, 52(3):226–34, 2001.

[8] Bin Tan and Fuchun Peng. Unsupervised query segmentation using generative language models and wikipedia. *International World Wide Web Conference*, 2008.

[9] Cayley Torgeson, Abdur Chowdhury, and Greg Pass. A picture of search. page 1, Hong Kong, junio 2006. ACM.