An Error Driven Approach to Query Segmentation

Wei Zhang[§][*], Yunbo Cao[‡], Chin-Yew Lin[‡], Jian Su[§], Chew-Lim Tan[†]
[§]Institute for Infocomm Research
[‡]Microsoft Research Asia
[†]National University of Singapore
{zhangw3,sujian}@i2r.a-star.edu.sg, {yunbo.cao,cyl}@microsoft.com, tancl@comp.nus.edu.sg

ABSTRACT

Query segmentation is the task of splitting a query into a sequence of non-overlapping segments that completely cover all tokens in the query. The majority of query segmentation methods are unsupervised. In this paper, we propose an error-driven approach to query segmentation (EDQS) with the help of search logs, which enables unsupervised training with guidance from the system-specific errors. In EDQS, we first detect the system's errors by examining the consistency among the segmentations of similar queries. Then, a model is trained by the detected errors to select the correct segmentation of a new query from the top-n outputs of the system. Our evaluation results show that EDQS can significantly boost the performance of state-of-the-art query segmentation methods on a publicly available data set.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Query Formulation*

Keywords

Query Segmentation, Search Log Mining, Error Driven

1. INTRODUCTION

Tokens (or words) of the queries for search engine are not independent or unordered symbols but rather ordered and structured words and phrases with syntactic relationships. Understanding the structure of a query is crucial for achieving better search performance. *Query segmentation* (QS), a process of splitting a query into a sequence of non-overlapping segments that completely cover all tokens in the query, aims to address these challenges. It requires that every segment rendered is a phrase or a semantic unit. More formally, let $q = [w_1, w_2, \cdots, w_n]$ denote a query consisting of n keywords. A segment $s = [w_i, \cdots, w_j](1 \leq i \leq j \leq n)$ is a subsequence of the query. A segmentation $S = [s_1|s_2|\cdots|s_K]$ for query q is then defined as a sequence of non-overlapping segments. '|' denotes a segmentation boundary. If we assume there is no order dependency of s, we can then treat S as a set $\{s_k\}_{k=1}^{K}$.

[*]Wei Zhang did this work when he was an intern at Microsoft Research Asia.

The majority of query segmentation methods are unsupervised, however, they are not as accurate as supervised methods due to the lack of guidance from labeled data. In this paper, we propose an error-driven approach to query segmentation (EDQS) with the help of search logs, which enables unsupervised training with guidance from the system-specific errors. EDQS first assumes the existence of a base segmentation system (hereafter referred to as '*base segmenter*') which is able to output top-n segmentations for any query. Then it tries to learn a *correction model* capable of replacing the rank-1 segmentation (if it is incorrect) with a 'rank-k'($k > 1$) segmentation (if one exists) from the output of the base segmenter. Our study on three state-of-the-art systems shows that for more than 25% of queries the correct segmentations are not ranked as top-1 but included in the top-5 results, which implies the potential of EDQS.

EDQS can be illustrated by the following flowchart. First, a query q is fed into a base segmenter (an implementation of any previous approach). As a result, a set of segmentations $\{S_i\}_{i=1}^{n}$ regarding q are generated. Subscript i denotes the rank of the corresponding segmentation. Next, $\{S_i\}_{i=1}^{n}$ are fed into a correction model. The correction model tries every possible replacement S_i $(i > 1)$ for the rank-1 segmentation S_1 (as indicated by the curved arrows). The trial ends with two possible results: (a) None of the replacements is valid (meaning that S_1 is correct); and (b) one segmentation S_i^* $(i^* > 1)$ is the most likely replacement and thus chosen as the final segmentation for q (e.g., the replacement indicated by the solid curve).

$$\begin{tikzcd}
 & S_1 & & S_1 \\
 & S_2 & & S_2 \arrow[u, bend right, dashed] \\
q \arrow[r, "\text{base segmenter}"] & S_3 \arrow[rr, "\text{correction model}"] & & S_3 \arrow[uu, bend right] \\
 & \cdots & & \cdots \\
 & S_n & & S_n \arrow[uuuu, bend right, dashed]
\end{tikzcd}$$

Next, we detail the keys to our proposal include: how to *automatically* detect the system-specific errors and then how to use the detected errors to learn a correction model.

2. SYSTEM-SPECIFIC ERROR DETECTION

Our method to detect the system-specific errors is motivated by the observation: *Queries with a similar intent tend to have consistent segmentation results.* We say that a set of queries have similar intents if and only if they lead to the same set of web documents (i.e., URLs). For example, when issuing to a web search engine any of the three queries in Table 1, we search for the same set of web pages which can provide 'free download of Adobe writer'. We denote such a set of queries as '*query intent set*'. More Formally, *Definition 1.* A *query intent set* Q^{INT} is a set of queries satisfying the following conditions:

a) $\bigcap_{q \in Q^{INT}} Urls(q) \neq \emptyset$;

b) $|Q^{INT}| > c$.

where $|Q^{INT}|$ denotes the number of elements in Q^{INT}, and c is set to 2 in our experiments.

Rank-1 Segmentation Result	Rank-2 Segmentation Result
download adobe | writer	**download | adobe writer**
free | adobe writer | download	free | adobe | writer | download
free | adobe writer	free adobe | writer

Table 1: Segmentation results for queries with a similar intent (Results in bold are considered 'correct'.)

For the queries in the same *query intent set*, naturally we wish to explain them in the same way and thus require that their segmentations be consistent with each other. We define the *consistency* $cst(S, S^{'})$ between segmentations S and $S^{'}$ as the number of segments they share, i.e.,

$$cst(S, S^{'}) = |S \cap S^{'}| \tag{1}$$

In Table 1, if we check only the 'rank-1' results, we observe that the segmentation 'download adobe | writer' disagrees with the other two, which is not what we expect to have. Instead, we expect to have the *bolded* segmentations in which none of the individual segments for one query disagrees with the segments for another query. Thus, we propose to detect the errors of rank-1 segmentation and select 'correct' segmentations from top-n segmentation results that are about m queries in the same *query intent sets* as follows,

$$(j_1^*, \cdots, j_m^*) = \underset{1 \le j_1, \cdots, j_m \le n}{\arg\max} \Big(\sum_{1 \le i, i^{'} \le m} \sum_{1 \le j^{'} \le n} cst(S_{ij_i}, S_{i^{'} j^{'}}) - \sum_{1 \le i \le m} cst(S_{ij_i}, S_{ij_i}) \Big) \tag{2}$$

Given one selected segmentation S_{ij_i}, the objective is to sum up the consistencies between itself and any of the rest $n * m - 1$ segmentations. Thus, by this objective, we choose the segmentations that are agreed with by most top-n segmentations. Finally, we can generate the instances to train the correction model as follows:

$$D_q = \begin{cases} \{(S_{q1} \mapsto S_{qj}, 0)\}_{j \neq 1} & \text{if } j^* = 1 \\ \{(S_{q1} \mapsto S_{qj^*}, 1)\} & \text{otherwise} \end{cases} \tag{3}$$

Note that this training set is generated based on a particular base segmenter, and thus the detected errors are system-specific.

3. CORRECTION MODEL

The decision of whether or not to correct S_{q1} to S_{qj} can be made by collectively considering one or multiple local transformations in the form of '$w_i w_{i+1} \mapsto w_i | w_{i+1}$' or '$w_i | w_{i+1} \mapsto w_i w_{i+1}$'. '$w_i w_{i+1} \mapsto w_i | w_{i+1}$' means that S_{q1} does not include a segment boundary between tokens w_i and w_{i+1} and S_{qj} does; '$w_i | w_{i+1} \mapsto w_i w_{i+1}$' means the reverse.

Let $T(S_{q1} \mapsto S_{qj})$ denote the set of all possible local transformations from S_{q1} to S_{qj} and $\mathbf{x}$ denote one element from the set (i.e., one local transformation). If we know the likelihood $f(\mathbf{x})$ of every individual transformation $\mathbf{x}$ being valid, the likelihood of replacing S_{q1} by S_{qj} can then be estimated as $\sum_{\mathbf{x} \in T(S_{q1} \mapsto S_{qj})} f(\mathbf{x})$.

The likelihood of a local transformation $\mathbf{x}$ being valid can be estimated with a binary classifier. We employ SVM as the classifier. Given an instance $\mathbf{x}$, SVM assigns a score to it based on $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$, where $\mathbf{w}$ denotes a weight vector and b denotes an intercept. Given a replacement $(S_{q1} \mapsto S_{qj}, y)$ where $y \in \{0, 1\}$, a set of labeled data for the binary classifier is prepared as: $\{\mathbf{x}, y\}_{\mathbf{x} \in T(S_{q1} \mapsto S_{qj})}$. By considering all the replacements in D_q, we will have a final training data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ for SVM.

On the basis of that, we can do the correction for a new query as follows: If for certain j $(j > 1)$ $\sum_{\mathbf{x} \in T(S_{q1} \mapsto S_{qj})} f(\mathbf{x}) > 0$, we will use the segmentation with $\underset{1<j \le n}{\arg\max} \sum_{\mathbf{x} \in T(S_{q1} \mapsto S_{qj})} f(\mathbf{x})$ as its index to replace the top-1 segmentation; Otherwise, we will keep using the top-1 segmentation. Table 2 describes the features for representing a local transformation $\mathbf{x}$.

Lexical	word w_i, word w_{i+1}, word pair $< w_i, w_{i+1} >$
MI	Mutual Information MI(w_i, w_{i+1}),MI(s_1, s_2)
Semantic	Freebase categories of s_1, s_2, s_3
Rank	j, the rank of the candidate segmentation
Direction	1, if "$w_i w_{i+1} \mapsto w_i \| w_{i+1}$"; 0, reverse.
Position^{left} Position^{right}	Number of words from the decision position to the beginning/end of query.

Table 2: The features for classifier. s_1,s_2 and s_3 denote the segment (including w_i, w_{i+1}, or both) of rank-1 segmentation or candidate segmentation.

4. EXPERIMENTS

Following Hagen et al. [3], we evaluate a QS system by query accuracy Acc^{qry}, break accuracy Acc^{brk} and segment F-score F^{sg}. We use two data sets as introduced in [1] and [2], denoted as **BW07** and **WQ10**. We mainly utilized three unsupervised systems as base segmenters. They are described in [2] , [3] and [4], denoted as **Base$^{\text{H-1}}$**, **Base$^{\text{H-2}}$** and **Base$^{\text{CN}}$** respectively. They can represent the state-of-the-art QS performance.

Table 3 reports the QS results. Comparing each pair of 'Base' and 'EDQS', we can see that EDQS proposed in this paper can be successfully spliced onto different base segmenters and significantly improves them over different data sets under the three evaluation metrics. ($p < 0.05$, **t-test**).

Data Set	Measure	**Base$^{\text{CN}}$**		**Base$^{\text{H-1}}$**		**Base$^{\text{H-2}}$**	
		Base	EDQS	Base	EDQS	Base	EDQS
BW07	Acc^{qry}	62.2	67.2	64.6	66.2	65.6	66.8
	Acc^{brk}	85.1	90.0	86.1	87.3	87.6	88.7
	F^{sg}	74.5	79.6	75.8	78.4	78.6	79.5
WQ10	Acc^{qry}	52.8	60.4	57.0	67.9	59.1	67.6
	Acc^{brk}	80.5	84.7	83.5	89.6	84.4	89.5
	F^{sg}	67.8	72.7	71.2	79.0	72.1	78.8

Table 3: Performance on query segmentation

5. REFERENCES

[1] S. Bergsma and Q. I. Wang. Learning noun phrase query segmentation. In *EMNLP-CoNLL*, 2007.

[2] M. Hagen, M. Potthast, B. Stein, and C. Bräutigam. The power of naive query segmentation. In *SIGIR*, 2010.

[3] M. Hagen, M. Potthast, B. Stein, and C. Bräutigam. Query segmentation revisited. In *WWW*, 2011.

[4] K. M. Risvik, T. Mikolajewski, and P. Boros. Query segmentation for web search. In *WWW*, 2003.