Learning a Replacement Model for Query Segmentation With Consistency in Search Logs

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 to split a query into a sequence of non-overlapping segments that completely cover all tokens in the query. The majority of methods are unsupervised, however, they are usually not as accurate as supervised methods due to the lack of guidance from labeled data. In this paper, we propose a new paradigm of *learning a replacement model with consistency* (LRMC), to enable unsupervised training with guidance from search log data. In LRMC, we first assume the existence of a base segmenter (an implementation of any existing approach). Then, we utilize a key observation that queries with a similar intent tend to have *consistent* segmentations, to automatically collect a set of labeled data from the outputs of the base segmenter by leveraging search log data. Finally, we employ the auto-collected data to train a replacement model for selecting the correct segmentation of a new query from the outputs of the base segmenter. The results show LRMC can improve state-of-the-art methods by an F-Score of around 7%.

1 Introduction

Nowadays *keyword queries* have been adopted as the de-facto query interface by most search engines. Query tokens 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. Such an understanding will also ease other search related applications such as query suggestion and rewriting, where one is able to work on semantic concepts instead of individual tokens. *Query segmentation* (**QS**), a process of splitting a query into a sequence of non-overlapping segments that completely cover all tokens, aims to address these challenges. It requires that every segment rendered is a phrase or a semantic unit. For example, given a query "download adobe writer", four different ways of segmentation are possible. The challenge is to determine which one is correct.

The majority of QS methods are unsupervised, however, they are not as accurate as supervised methods due to lack of guidance from labeled data. On the other hand, supervised models suffer from the problems: (1) new phrases/words are introduced on the web daily, which quickly invalidate static supervised models trained on a certain manually labeled set; (2) it is not feasible to develop a set of labeled data covering all domains on the web. In this paper, we propose a paradigm of *learning a replacement model with consistency* (LRMC), to enable unsupervised training and it improves various unsupervised QS systems.

LRMC first assumes the existence of a base segmentation system (hereafter referred to as '*base segmenter*') which can output top-n segmentations for any query. Then it tries to learn a *replacement model* capable of selecting the correct segmentation of a new query (if one exists) from the output of the base segmenter. Our study on three state-of-the-art systems (Section 5.2) shows that for more than 35% of queries the correct segmentations are not ranked as top-1 but included in the top-5 results of the base segmenter, which implies the potential of LRMC. The keys to our proposal include: (a) how to *automatically* acquire labeled data (i.e., for a query in the labeled data, what its correct segmentation is) and then (b) how to use the labeled data to learn the replacement model.

Our method for the automatic acquisition of the labeled data is motivated by the observation: *Queries with a similar intent tend to have consistent segmentation results.* In this paper, 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., clicks). For example, when issuing to a web

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

International Joint Conference on Natural Language Processing, pages 28–36,
Nagoya, Japan, 14-18 October 2013.

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

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

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*'. 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 say that q_1 and q_2 are inconsistent in segmentation if there exist more than one common subsequence of tokens having different segment boundaries. In Table 1, we also include the top-2 segmentation results that can possibly be generated by any base segmenter. If we check only the 'rank-1' results, we observe that the segmentation 'download adobe | writer' disagrees with the other two. This means that we interpret the same sequence of tokens differently for different queries with a same intent, 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. In this paper, we propose two methods for selecting such correct segmentations from top-n segmentation results that are about the same *query intent sets*. With these methods, we can automatically build up a training data set, which allows us to train a reliable model.

The replacement model concerns about *whether or not a 'rank-1' segmentation S^a generated by a base segmenter should be replaced by a 'rank-k' ($k > 1$) segmentation S^b*. The decision of the replacement can be made by collectively considering one or multiple local transformations in the form of '$w_iw_{i+1} \mapsto w_i|w_{i+1}$' or '$w_i|w_{i+1} \mapsto w_iw_{i+1}$'. '$w_iw_{i+1} \mapsto w_i|w_{i+1}$' means that S^a does not include a segment boundary between tokens w_i and w_{i+1} and S^b does; Similarly, '$w_i|w_{i+1} \mapsto w_iw_{i+1}$' means the reverse. For example, for the first query in Table 1, we can have the local transformations 'download adobe $\mapsto$ download | adobe' and 'adobe | writer $\mapsto$ adobe writer'. The proposed model estimates the score of every local transformation using a binary classifier and then aggregates the individual scores to reach its final decision.

We conduct extensive experiments using two public data sets. The results show that (a) our method for automatically constructing a set of labeled data with a base segmenter and a set of query intent sets as inputs is effective, capable of discovering correct segmentations missed by the evaluated base segmenters for more than 20% of queries (See **M2** in terms of Acc^{qry} in Table 3); and (b) our replacement model benefits existing QS approaches and boosts their performance significantly (e.g. the improvement of $> 7\%$ F-Score on the data WQ10-Majority in Table 4).

We summarize our contributions as follows: (1) on the basis of the observation that queries with a similar intent tend to have consistent segmentations, we propose a method for automatically collecting from search log data a set of labeled data for QS. The method first groups queries in search log data into what we call a 'query intent set' and then select correct segmentations by examining the consistency among segmentations for the queries in the same 'query intent sets'. (2) With the automatically-collected data, we develop a 'replacement model' for the purpose of checking whether or not a 'rank-1' segmentation generated by a base segmenter should be replaced by a 'rank-k' ($k > 1$) segmentation. (3) We conduct extensive experiments with two publicly available data sets and show that our proposal can effectively boost the performance of state-of-the-art systems (Hagen et al., 2010; Hagen et al., 2011).

2 Related Work

Bergsma and Wang (2007) considered the decision to segment or not between each pair of adjacent words as a binary classification problem. Guo et al. (2008), Yu and Shi (2009), and Kiseleva et al. (2010) used methods based on CRF. As the cost of obtaining labeled data is high, they are usually not feasible to develop a set of labeled data covering all the domains on the web and then train a scalable QS model for web search.

The work for web-scale QS are usually unsupervised and utilized various statistics such as mutual information (MI) and frequency count collected from various sources such as web data, query logs, and etc (Risvik et al., 2003; Jones et al., 2006; Huang et al., 2010; Zhang et al., 2009). Li et al. (2011) also used the language model estimated

from click-through documents to backoff the generating process of QS. Tan and Peng (2008) used n-gram frequencies from a large web corpus as well as Wikipedia. Hagen et al. (2010) showed that the raw n-gram could be exploited with an appropriate normalization scheme and achieved surprisingly good accuracy. Later, they enriched the work by including the use of Wikipedia (Hagen et al., 2011). In our evaluation, we compare our proposal with the last two work which represent state-of-the-art.

Our proposal is orthogonal to all the above approaches. LRMC assumes the existence of a base segmenter (an implementation of any above approaches) and it focuses on how to leverage search log data to learn a replacement model for improving the output of base segmenters.

3 Problem Settings

QS. 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 $\{s_k\}_{k=1}^{K}$.

Query Intent. There exist many definitions on query intent. In this paper we introduce an operational definition on query intent.

Definition 1 *The* query intent*(s) of a query q is defined as the set of URLs ($Urls(q)$) which are clicked for q by users of a web search engine.*

Because most queries are ambiguous or multi-faceted (Clarke et al., 2009), we manage to restrict the number of intents into one or a few by grouping more queries together, which leads to the definition of 'query intent set'.

Definition 2 *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 a parameter to control how specific a query intent is; a larger value for c usually means that the query intent is more specific and thus less ambiguous.

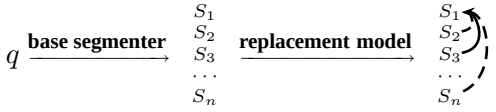
Query intent sets used in our experiments will be detailed in Section 5.1.

4 Our Proposal

4.1 Overview of the Proposed Paradigm

First, the paradigm LRMC assumes the existence of a base segmenter that is able to output top-n segmentations for any query. Then it tries to learn a *replacement model* capable of replacing the rank-1 segmentation generated by this base segmenter with one rank-k ($k > 1$) segmentation.

LRMC can be illustrated by the following flowchart. First, a query q is fed into a base segmenter. 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 replacement model. The replacement 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 (S_1 cannot be replaced); 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 of the solid curve).



LRMC is motivated by the following observation: for most cases, the correct segmentation for a query is included in its top-n segmentation results already. Usually, there are not that many likely segmentations for a query and thus correct segmentations cannot be ranked too low by a base segmenter. For example, for any base segmenter in our experiment, more than 93% of queries can have a segmentation that is agreed upon by at least one of the annotators in its top-5 results. Given this observation, what we have to do is not to generate or propose a new segmentation, but to tell which segmentation is correct in the top-n results.

Next, we detail how the replacement model is learned. Specifically, we first introduce how we automatically extract from search log data a set of labeled data with 'consistency' as a guidance and then explain how a 'replacement model' can be learned from this data set.

4.2 Consistency as Supervision

Assume that we have a *query intent set* $Q^{INT} = \{q_i\}_{i=1}^{m}$. With a base segmenter, we generate the top-n segmentation results $\{S_{ij}\}$ $(1 \leq j \leq n)$ for

each query q_i, which forms the following matrix:

$$S_{Q^{INT}} = \begin{pmatrix} S_{11} & \underline{S_{12}} & S_{13} & \cdots & S_{1n} \\ S_{21} & \underline{S_{22}} & S_{23} & \cdots & S_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \underline{S_{m1}} & S_{m2} & S_{m3} & \cdots & S_{mn} \end{pmatrix} \begin{matrix} \leftarrow & q_1 \\ \leftarrow & q_2 \\ & \cdots \\ \leftarrow & q_m \end{matrix}$$

What we manage to achieve is to collect a set of 'labeled' data from $\{S_{ij}\}$. In the 'labeled' data, each query q_i has only one segmentation S_{ij^*} ($S_{ij^*} \in \{S_{ij}\}_{j=1}^{n}$), which we consider 'correct'. We make use of two types of strategies to choose the 'correct' segmentations from $\mathbf{S_{Q^{INT}}}$ (e.g., those underlined ones in $\mathbf{S_{Q^{INT}}}$, namely $S_{12}, S_{22}, \cdots, S_{m1}$).

Before explaining the two strategies, let us first introduce how we measure the consistency between two segmentations. The *consistency* $cst(S, S^{'})$ between segmentations S and $S^{'}$ is defined as the number of segments they share, i.e.,

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

The first strategy (**M1**) that we use as 'supervision' for the acquisition of the labeled data is as follows: The correct segmentations for the m queries in the same *query intent set* should be very consistent with or similar to each other although the segmentations cannot be exactly the same. Thus, the correct segmentations can be chosen with the objective function:

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

where j_i^* denote the index (or rank) of the correct segmentation for query q_i.

The other strategy (**M2**) is on the basis of the observation: Although at most only one top-n segmentation can be correct for a query, most segmentations are not totally incorrect, i.e., they include some correct segments while having some incorrect segments as well. Thus, those incorrect segmentations also provide some clues about what can be correct. In addition, as the choices for 'incorrect segment' are usually more than those for correct segment, it is relatively hard for incorrect segments to converge to a few. As a result, a correct segment should be more popular than any one single incorrect segment. Given this discussion, we can have the second objective function:

$$(j_1^*, \cdots, j_m^*) = \underset{1 \le j_1, \cdots, j_m \le n}{\arg\max} \Big(\sum_{1 \le i, i^{'} \le m} \sum_{1 \le j^{'} \le n} cst(S_{ij_i}, S_{i^{'}j^{'}}) - \sum_{1 \le i \le m} cst(S_{ij_i}, S_{ij_i}) \Big) \tag{3}$$

Note that $cst(S_{ij_i}, S_{ij_i}) = |S_{ij_i}|$. Given one selected segmentation S_{ij_i}, the objective is to sum up the consistencies between itself and any of the rest in matrix $S_{Q^{INT}}$. Thus, by this objective, we choose the segmentations whose segments are agreed with by most top-n segmentations.

Both strategies assume that correct segments are more popular than incorrect segments in the top-n output of one reasonably-performing base segmenter. Both strategies will fail if the assumption is not true. Our experiments in Section 5.2, in which both strategies are able to find more correct segmentations than the base segmenters, can be seen as a support for the assumption.

4.3 Replacement Model

The replacement model is to tell whether or not a segmentation ranked as top-1 by a base segmenter should be replaced by another segmentation with a rank of j ($1 < j \le n$). For example, we have a query q whose top-n segmentations are $\{S_{qj}\}_{j=1}^{n}$. Then, the input of the replacement model will be a possible replacement $S_{q1} \mapsto S_{qj}$ ($j > 1$) and the output will be a label '1' or '0'. Label '1' means S_{q1} should be replaced by S_{qj} and '0' means 'not'.

With that in mind, we can then make use of '*consistency*' to create a labeled data set. For example, if query q belongs to a query intent set Q^{INT} and its correct segmentation chosen by the objective (2) or (3) is S_{qj^*}, we can generate the labeled instance(s) as follows:

$$D_q = \begin{cases} \{(S_{q1} \mapsto S_{qj}, 0)\}_{j \ne 1} & \text{if } j^* = 1 \\ \{(S_{q1} \mapsto S_{qj^*}, 1)\} & \text{otherwise} \end{cases} \tag{4}$$

By combining all such data sets together, we then have the final labeled data set $D = \underset{q}{\cup} D_q$. Note that query q can come from multiple query intent sets (not just one single set).

Next, we explain how to use the above training data to learn a replacement model.

The decision of whether or not to do the replacement of $S_{q1} \mapsto S_{qj}$ 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 score 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, 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 replacement 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 $\arg\max_{1<j\leq n} \sum_{\mathbf{x}\in T(S_{q1}\mapsto S_{qj})} f(\mathbf{x})$ as its index to replace the top-1 segmentation; Otherwise, we will keep using the top-1 segmentation.

4.4 Learning Features

In this section, we describe the features for representing a local transformation $\mathbf{x}$, which is in the form of either '$w_{i_0}w_{i_0+1} \mapsto w_{i_0}|w_{i_0+1}$' or '$w_{i_0}|w_{i_0+1} \mapsto w_{i_0}w_{i_0+1}$'. We utilize four categories of features which are possible indicators of a transformation, representing a variety of information such as lexical, syntactic, semantic and etc.

Contextual Features: Lexical. The left and right tokens around the decision position, w_{i_0} and w_{i_0+1}, are a good signal of the transformation. In the example of "google desktop | download", the token 'download' is separated from its left neighbor. Such common query tokens in the training data with the property of usually being separated from or being connected to its left/right neighbor can help predict new transformations (e.g. "adobe writer download ↦ adobe writer | download"). On the basis of this observation, we adopt the left token w_{i_0} and the right token w_{i_0+1} as the features for representing a transformation $\mathbf{x}$. Furthermore, sometimes one word alone can not perfectly characterize a transformation. For example, to reject the transformation "diet plan ↦ diet | plan", we have to use the token bigram <diet plan>. Thus, we include all the token bigrams in the form of $< w_{i_0}\ w_{i_0+1} >$ as features as well. As we all know, lexical features usually suffer from a data sparseness issue when used in various tasks (Bagga and Baldwin, 1998; Sriram et al., 2010). Fortunately, the web-scale training data we collect from *query intent sets* (Section 4.2) enables us to have a good coverage of lexical features.

Contextual Features: POS Tag. Bergsma et al. (2007) show that part-of-speech (POS) tags are useful in their segmentation classification. We also exploit the POS tag pair of w_{i_0} and w_{i_0+1} as features. For example, intuitively, "NN NN ↦ NN | NN" is more likely to occur than "JJ NN ↦ JJ | NN". The POS tags that we consider include all types of POS tags. Note that this is different from Bergsma et al. (2007). As their segmentation model only takes care of noun phrase queries, their POS tags are restricted to determiners, adjectives, and nouns. The POS tagger by (Roth and Zelenko, 1998) is used in this paper.

Mutual Information (MI). Following previous work (Section 2), we also adopt MI between w_{i_0} and w_{i_0+1} as our feature. The work (Bergsma and Wang, 2007) also considered the case of a noun phrase with multiple modifiers (e.g. "female bus driver"). To make the segmentation decision between 'female' and 'bus', MI('female', 'driver') is more suitable to represent the information of not separating them than MI('female', 'bus'). Thus, we also incorporate $MI(w_{i_0-1}, w_{i_0+1})$ and $MI(w_{i_0}, w_{i_0+2})$ into our feature set.

Most previous work on QS only can use word-based MI as introduced above. However, in some cases, the MI between tokens can not provide sufficient information for a segmentation decision. For instance, assume that we have the following two queries with their correct segmentations: (1) "download | call of duty | free"; (2) "duty free | shops | sfo". Only using the token-based mutual information MI('duty', 'free') can not discriminate the two queries from each other and thus can not give different segmentations for 'duty free' in the two queries. In our work, as the query has been segmented by a base segmenter, we propose to also use the segment-based MI. In the 'duty free' example, MI('call of duty', 'free') will be incorporated for the transformation decision related to "download | call of duty free ↦ download | call of duty | free", where the token-based MI('duty', 'free') does not work.

Semantic Features. We define the semantic features on the basis of segments. For a transformation '$w_{i_0}w_{i_0+1} \mapsto w_{i_0}|w_{i_0+1}$' , let us denote the

segment including $w_{i_0}w_{i_0+1}$ (in the first segmentation) as s_1 and denote the segments including w_{i_0} and w_{i_0+1} separately (in the second segmentation) as s_2 and s_3, respectively. To obtain semantic labels for the above three segments, we make use of a web-scale knowledge base of entities, namely Freebase (Bollacker et al., 2008). First, we map the three segments to the Freebase entities by string matching, and then use the names or aliases of the associated categories of the mapped entities as their semantic labels. Finally, the semantic labels for s_1, s_2 and s_3 are used as features. Due to the ambiguity, a phrase in a query may be mistakenly linked to a certain entity in the knowledge base. Thus, the semantic features include some noises. However, even with noises such features can still contribute to QS. To illustrate how the semantic features work, consider the query with the assigned semantic label as follows,

[history of the]$_{\text{NULL}}$ [search engine]$_{\text{computer software genre}}$

As pointed out by (Tan et al., 2008), QS approaches which are only based on statistical information (e.g. MI and frequencies of n-grams) collected from the Web, cannot guarantee that the resulting segments are meaningful ones. For the query 'history of the search engine', a possible segmentation is 'history of the | search engine', as both 'history of the' and 'search engine' occur on the web frequently. In contrast, semantic information can distinguish 'search engine' from 'history of the', as 'history of the' is labeled as NULL and 'search engine' is labeled as 'computer software genre'. Moreover, the learning algorithm can also learn some implicit relations between transformations and semantic labels, e.g. some particular combination of labels for s_1, s_2 and s_3 may often trigger or prevent a transformation.

Rank, Direction and Position Features. Table 2 shows the values of these features. The rank feature is designed to distinguish among the different segmentation rankings of a base segmenter. For example, this feature can capture the intuition that for a good base segmenter, top ranked segmentations should have more of a chance to be selected. The direction feature is used to distinguish the two kinds of transformations: '$w_iw_{i+1} \mapsto w_i|w_{i+1}$' and '$w_i|w_{i+1} \mapsto w_iw_{i+1}$'. The position feature considers decision positions, as transformations in different positions may have different chances.

Rank	j, the rank of the segmentation to which we transform the top-1 segmentation.
Direction	1, if "$w_iw_{i+1} \mapsto w_i\|w_{i+1}$"; 0, reverse.
Position^{left} Position^{right}	Number of words from the decision position to the beginning/end of query.

Table 2: The 'rank', 'direction' and 'position' features

5 Experiments

5.1 Experimental Setup

Following Hagen et al. (2011), we evaluate a QS system at three levels: **Query Level:**

$$Acc^{qry} = \frac{\#\text{correctly segmented queries}}{\#\text{queries in the evaluation data set}} \quad (5)$$

Break Level. The decision of break is whether or not to insert a segment boundary between two tokens in the query. The break-level accuracy (Acc^{brk}) is defined as the proportion of the correctly-made decisions out of all such decisions.

Segment Level. Let Q^{eval} denote the set of queries. S_q^{sys} is the segmentation generated by a system and S_q^{eval} is given by a human. Then,

$$P^{sg} = \sum_{q \in Q^{eval}} \frac{|S_q^{sys} \cap S_q^{eval}|}{|S_q^{sys}|}$$
$$R^{sg} = \sum_{q \in Q^{eval}} \frac{|S_q^{sys} \cap S_q^{eval}|}{|S_q^{eval}|} \quad F^{sg} = \frac{2 \cdot P^{sg} \cdot R^{sg}}{P^{seg} + R^{sg}} \quad (6)$$

We use two data sets as introduced in (Bergsma and Wang, 2007) and (Hagen et al., 2010), denoted as 'Bergsma-Wang-07' (**BW07**) and 'Webis-QSeC-10' (**WQ10**). BW07 includes 500 test queries which all were noun phrase queries. Each query was segmented manually by three annotators (denoted as annotator A, B, and C) respectively. For 44% of the queries, all three annotators agree on the segmentations. Such an agreement between annotators cannot be considered as 'strong', which to some extent implies that human annotations may not be so reliable when used for training a segmentation model capable of consistently working over different queries. WQ10 includes 4,850 queries. Each query can be any type of query, not necessarily a noun phrase query. Each query was annotated by ten annotators.

We made use of the mining method in (Hu et al., 2011) for collecting the *query intent sets*. With the search log data and clicks (Apr 1, 2009-Mar 31, 2010) as input, we finally obtained 9,412,308 query intent sets, which totally include 30,902,284 unique queries. The similar queries in each set share more than 10 clicks. Each set includes 2~11 queries. We denote this data set as **QSet**. Note

that this data set does not have any annotations. Thus, we also tried to construct another data set (denoted as **QSet**^{ann}) by intersecting **QSet** with WQ10. **QSet**^{ann} includes 1,554 queries. Every query in **QSet**^{ann} is then associated with the human annotations from WQ10 and linked to at least one query intent set in **QSet**.

As each query has more than one segmentation due to different annotators, we select segmentation as our reference under two schemes: '**Majority**' where the segmentations agreed upon by a majority of the annotators are chosen as the reference, and '**Best**' where the annotated segmentations that maximize the break accuracy Acc^{brk} of the evaluated segmenter are chosen as the reference.

We mainly utilized three unsupervised systems as base segmenters. They are described in (Hagen et al., 2010) , (Hagen et al., 2011) and (Risvik et al., 2003), 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. For example, **Base**$^{\text{H-2}}$ on on data BW07(A) achieves $69.2\% F^{sg}$ which slightly outperforms the recent unsupervised system (Li et al., 2011) ($69.0\% F^{sg}$).

As we focus on web QS, we did not compare LRMC with supervised methods which are only designed for one particular domain. For example, Yu et al. (2009)'s method is for queries of relational databases. The work (Bergsma and Wang, 2007) and the supervised stage of (Bendersky et al., 2009) are only for noun-phrases.

5.2 Consistency as Weak Supervision

LRMC relies on a training data which is automatically collected with the help of query intent sets. Thus, in this section, we evaluate the training set collected by **M1** and **M2** (Section 4.2).

In the experiments, we first applied a base segmenter to the queries in **QSet** and then managed to choose one segmentation as correct from the output for each query with either **M1** or **M2**. Last, we evaluated the new output by checking only the segmentations for the queries in subset **QSet**^{ann}. Some queries in **QSet**^{ann} may belong to different intent-sets and in each intent-set may have different segmentation labels as 'correct'. In our evaluation, we randomly selected one of them as the final label. Besides, we also included an ideal method **Oracle** by which the correct segmentation can always be identified and used as the new output if the segmentation exists in the top-n results of the base segmenter. Note that **Oracle** is an upper-bound result obtained by directly matching with human's annotation and cannot be applied to query intent sets **QSet** for collecting labeled data. Table 3 provides the results, where top-k ($1 \leq k \leq 5$) means that the input to **M1**/**M2** is the top-k results of the corresponding base segmenter. Note that the top-1 results are the performance of the base segmenters.

By checking the results of **Oracle**, we can find that for every base segmenter, more than 35% of the correct segmentations in the top-5 results are not covered by the top-1 results (in terms of Acc^{qry}). In addition, around 90% correct segmentation boundaries (Acc^{brk}) are included in the top-5 results of the base segmenters. These findings indicate the feasibility of our replacement model, which tries to replace a rank-1 segmentation by a rank-k ($k > 1$) segmentation.

From the table, we also see that both **M1** and **M2** are able to significantly perform better than the base segmenters do ($p < 0.05$, t-test). This can be observed through all the measures. For example, using Acc^{qry} as the evaluation metric, the percentage of the correct segmentations that **M2** discovers more than the base segmenters do ranges from 20.2% to 24.6%. These improvements prove the underlying assumption that queries with a similar intent tend to have consistent segmentation. Besides, we can see that **M2** can reach a satisfied performance to collect the labeled data. For example, break-level accuracy Acc^{brk} can reach 80%.

Table 3 also shows that M1 and M2 perform best by using top 3 or 4 results from base segmenter. This finding indicates that our framework should work with a reasonable base segmenter.

By comparing the results generated by **M1** and **M2** with all three measures, we see that **M2** consistently performs better than **M1**, and the difference is statistical significant ($p < 0.05$, t-test). This tells us that consistency should be calculated with all the top-n segmentations rather than with only the selected segmentations.

5.3 Query Segmentation

In this section, we investigate the effectiveness of our LRMC which is a combination of data collecting method and replacement model.

In the experiments, we first made use of **M2** to automatically collect the training data from the query intent sets **QSet**. During the process of col-

Segmenter	Rank	Oracle			M1			M2		
		Acc^{qry}	Acc^{brk}	F^{sg}	Acc^{qry}	Acc^{brk}	F^{sg}	Acc^{qry}	Acc^{brk}	F^{sg}
Base^{CN}	Top-1	38.7	67.9	51.7	38.7	67.9	51.7	38.7	67.9	51.7
	Top-2	46.1	76.1	60.8	42.4	70.0	55.6	47.0	76.4	56.3
	Top-3	67.3	85.6	74.9	**58.8**	**81.3**	**68.3**	58.8	81.0	68.1
	Top-4	73.3	88.5	79.9	58.3	79.6	67.6	**58.9**	**81.4**	**68.3**
	Top-5	**75.2**	**89.5**	**81.5**	44.6	72.1	57.6	59.6	61.0	60.2
Base^{H-1}	Top-1	42.9	69.7	53.8	42.9	69.7	53.8	42.9	69.7	53.8
	Top-2	59.0	81.7	68.9	46.0	75.9	60.6	47.2	77.2	61.7
	Top-3	72.5	87.8	78.5	63.2	83.0	70.8	**65.3**	82.5	**72.0**
	Top-4	75.2	89.4	81.2	**64.3**	**83.3**	**71.6**	64.3	**83.5**	71.8
	Top-5	**77.9**	**90.6**	**83.2**	59.0	81.7	68.7	63.0	82.1	70.0
Base^{H-2}	Top-1	39.6	68.3	52.2	39.6	68.3	52.2	39.6	68.3	52.2
	Top-2	51.6	78.5	64.2	45.6	74.0	59.4	45.4	73.5	59.2
	Top-3	69.4	86.4	76.3	**61.0**	**80.1**	**69.4**	64.2	**83.4**	**71.2**
	Top-4	74.1	88.9	80.5	59.2	78.9	69.0	63.1	82.2	70.0
	Top-5	**76.7**	**90.1**	**82.5**	59.8	78.0	67.2	**67.1**	66.3	66.5

Table 3: Consistency as weak supervision on **QSet**^{ann} (Majority)

lecting the data, we took only the top-3 segmentations as input. The training data set collected by **M2** contains around 45 million instances (pairs of segmentations). Then, all this labeled data is used to train the replacement model as introduced in Section 4.3. We made use of LIBSVM (Chang and Lin, 2011) and a linear kernel in our experiment. Finally, we applied the learned replacement models to the evaluation data sets BW07 and WQ10.

Table 4 reports the QS results[1]. Following previous work (Bergsma and Wang, 2007; Hagen et al., 2011), we report four groups of results with the data BW07. In each of the first three groups, only the reference segmentations from annotator A, B or C are used. The fourth group is '**Best**' of BW07. We also report two groups of results ('**Majority**' and '**Best**') with the data WQ10. Comparing each pair of 'Base' and 'LRMC', we can see that LRMC 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 Acc^{qry}, Acc^{brk} and F^{sg}. ($p < 0.05$**, t-test**). Especially, the state-of-the-art systems **Base^{H-1}** and **Base^{H-2}** have been significantly improved by LRMC. The improvements prove that the automatically-collected labeled data can guide QS and our replacement model can take advantage of the data.

Data Set	Measure	Base^{CN}		Base^{H-1}		Base^{H-2}	
		Base	LRMC	Base	LRMC	Base	LRMC
BW07 (A)	Acc^{qry}	53.4	55.3	55.2	56.7	53.8	55.4
	Acc^{brk}	79.3	81.4	80.2	81.9	79.5	81.7
	F^{sg}	66.5	69.6	67.5	70.2	66.7	69.8
BW07 (B)	Acc^{qry}	37.4	40.2	39.8	41.1	37.8	39.8
	Acc^{brk}	73.7	74.9	74.7	76.4	73.8	75.4
	F^{sg}	54.3	58.3	55.6	58.7	54.5	58.1
BW07 (C)	Acc^{qry}	41.6	44.2	43.8	46.7	42.0	46.9
	Acc^{brk}	74.1	75.2	75.0	78.6	74.2	78.6
	F^{sg}	56.9	60.4	58.0	62.3	57.1	62.4
BW07 (Best)	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 (Majority)	Acc^{qry}	30.0	38.5	31.8	40.1	30.3	39.8
	Acc^{brk}	65.3	72.1	66.2	74.0	65.5	73.1
	F^{sg}	47.5	55.1	48.5	55.8	47.7	55.6
WQ10 (Best)	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 4: Performance on query segmentation

6 Conclusions and Future Work

We have proposed a paradigm LRMC for QS. LRMC assumes the existence of a base segmenter and then learns how to select correct segmentations from the output of the base segmenter. The replacement model is trained by a labeled data set which can be automatically collected from *query intent sets*, instead of relying on any human annotation. There exist two interesting directions for future work: (1) we observe that there is still a big gap in performance between the proposed methods and **Oracle**. According to our analysis, most of the gap is caused by that the incorrect segmentations for some similar queries also happen to have a high consistency when measured by either proposed strategy. Thus, it is worth studying other methods that can address such performance gap. (2) we would like to further explore the concept of query intent sets. In this paper, we assume that similar intent queries tend to have similar segmentations. A reasonable next step is to explore the idea that similar intent queries tend to have similar labels, which can be useful for the task of tagging query segments with semantic labels.

[1]Note that the results for the base segmenters **Base^{H-1}** and **Base^{H-2}** are not exactly same as those reported in (Hagen et al., 2011) although they are very close. For example, **Base^{H-1}** and **Base^{H-2}** on WQ10 achieve 73.4% F^{sg} and 74.2% F^{sg} in the original paper. Ours are 71.2% and 72.1%. The reasons are as follows: For BW07, they used a cleaned version of the data set; for WQ10, they released just a subset of the data used in their experiments.

References

A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *COLING*, 1998.

M. Bendersky, W. B. Croft, and D. A. Smith. Two-stage query segmentation for information retrieval. In *SIGIR*, 2009.

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

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. *Intelligent Systems and Technology*, 2011.

C. L. A. Clarke, N. Craswell, and I. Soboroff. Overview of the trec 2009 web track. In *TREC*, 2009.

J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *SIGIR*, 2008.

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

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

Y. Hu, Y. Qian, H. Li, D. Jiang, J. Pei, and Q. Zheng. Mining query subtopics from search log data. In *WWW*, 2011.

J. Huang, J. Gao, J. Miao, X. Li, K. Wang, F. Behr, and C. L. Giles. Exploring web scale language models for search query processing. In *WWW*, 2010.

R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW*, 2006.

J. Kiseleva, Q. Guo, E. Agichtein, D. Billsus, and W. Chai. Unsupervised query segmentation using click data: preliminary results. In *WWW*, 2010.

Y. Li, B.-J. P. Hsu, C. Zhai, and K. Wang. Unsupervised query segmentation using clickthrough for information retrieval. In *SIGIR*, 2011.

N. Mishra, R. S. Roy, N. Ganguly, S. Laxman, and M. Choudhury. Unsupervised query segmentation using only query logs. In *WWW*, 2011.

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

D. Roth and D. Zelenko. Part of speech tagging using a network of linear separators. In *Coling-Acl, The 17th International Conference on Computational Linguistics*, 1998.

B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas. Short text classification in twitter to improve information filtering. In *SIGIR*, 2010.

B. Tan and F. Peng. Unsupervised query segmentation using generative language models and wikipedia. In *WWW*, 2008.

D. Wu, Y. Zhang, and T. Liu. Unsupervised query segmentation using monolingual word alignment method. *Comp. and Info. Science*, 5(1), 2012.

X. Yu and H. Shi. Query segmentation using conditional random fields. In *KEYS*, 2009.

C. Zhang, N. Sun, X. Hu, T. Huang, and T.-S. Chua. Query segmentation based on eigenspace similarity. In *ACL-IJCNLP*, 2009.