Unsupervised Query Segmentation Using Click Data: Preliminary Results

Julia Kiseleva Qi Guo Eugene Agichtein
Emory University
{jkisele,qguo3,eugene}@mathcs.emory.edu

Daniel Billsus Wei Chai
Shopping.com, an eBay company
{dbillsus,wechai}@shopping.com

ABSTRACT

We describe preliminary results of experiments with an unsupervised framework for query segmentation, transforming keyword queries into structured queries. The resulting queries can be used to more accurately search product databases, and potentially improve result presentation and query suggestion. The key to developing an accurate and scalable system for this task is to train a query segmentation or attribute detection system over labeled data, which can be acquired automatically from query and click-through logs. The main contribution of our work is a new method to automatically acquire such training data – resulting in significantly higher segmentation performance, compared to previously reported methods.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval

General Terms

Performance, Design, Experimentation.

Keywords

Query segmentation, attribute extraction, structured queries

1. INTRODUCTION

This work focuses on the problem of detecting and labeling product attribute values in e-commerce keyword queries. Identifying attribute values in a keyword query would enable structured querying of product databases, more effective ranking and filtering of the results, and potentially improving the result presentation. The main contribution of this work is an *unsupervised* approach to this problem that trains the attribute extraction system with data derived from product click-through information. The key idea is to automatically and robustly align the query terms to attribute terms via click data, resolve ambiguities using frequency and similarity statistics, and then use the resulting automatically generated alignments to train the text segmentation component of an information extraction system.

This unsupervised approach has multiple advantages over previous supervised and semi-supervised methods:

- Our method requires no manual labeling, which can be time consuming to obtain for a large number of domains.
- Our method can be constantly updated to reflect changes in user interest and product databases.
- We can cleanly trade-off different performance characteristics by controlling the parameters of the matching (automatic labeling) process.

Our results, based on a sample of data from collected from the Shopping.com search logs, demonstrate the accuracy and scalability of our approach.

2. SYSTEM OVERVIEW

The overall data flow of our system is illustrated in Figure 1 (a-b). Step (a) illustrates the training process. Step (b) illustrates the runtime process. Our system has three main components:

- **Automatic labeling**: automatically label query attributes using click-through data (Section 3).
- **Training a segmentation model**: using the automatically derived training data to train a segmentation system that can attach attribute names to query tokens (Section 4).
- **Segmenting queries using the learned segmentation model**: predicting token labels for new queries (Sections 5).

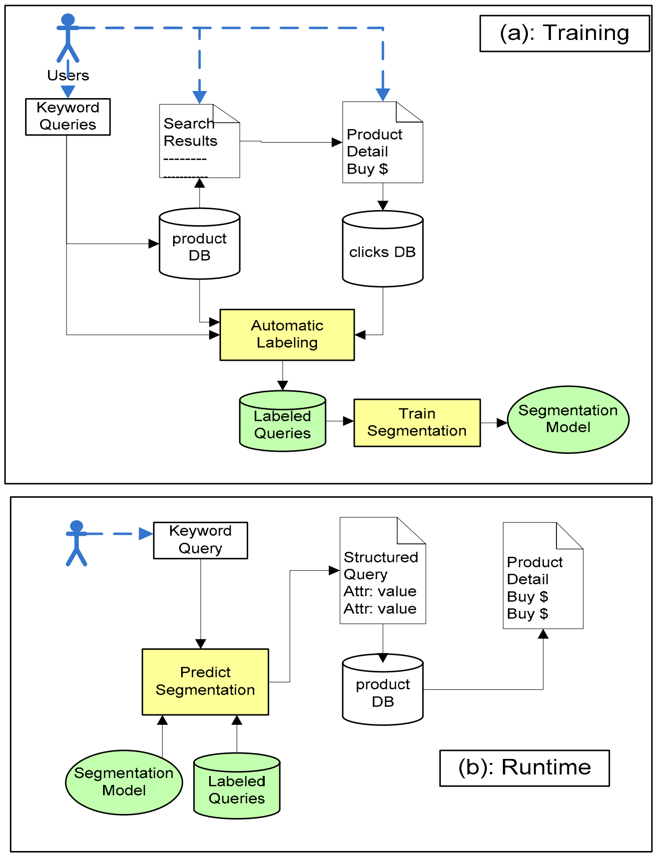

Figure 1: System Overview: training based on automatically labeled data (a), and run-time prediction using the trained model and previously labeled queries (b).

3. AUTOMATIC TOKEN LABELING

Our approach applies to the following problem setup. Shoppers run queries against a product search engine (e.g. '*sony vaio*','*4gb laptop*', etc). These products have a product *name* and *textual description*, as well as a set of structured attributes represented as name-value pairs (e.g. *type:* 'Notebook', *model*: 'Vaio', *brand*: 'Sony', *processor speed*: '2.0ghz', etc.). The prediction task is to automatically label each query token with a corresponding product attribute. To generate training data automatically, we leverage user interactions with the search engine. That is, we record clicks on individual products in response to a query, assuming that users click more frequently on products that are relevant to the query. We compare two token labeling methods:

- **Baseline.** As a baseline matching algorithm, we use the method described in reference [1]. This method assigns an attribute name to a query token if the attribute value from a clicked product contains the token *unambiguously*: that is, only one attribute value in the product contains the token. The main disadvantage of this method is that many tokens appear in multiple attribute values, and thus remain unlabeled.
- **Similarity:** As our method, we use cosine similarity to quantify the match degree between a token and an attribute value string. The weights of the terms are computed using a modified tf*idf weighting scheme. Our modification is to define a "document" as the combination of all tokens for each attribute, combined across all the clicked products. For example, a *Brand* "document" will contain all observed tokens and their counts within the *Brand* attribute across all laptops, e.g., <'dell':14, 'lenovo':9, 'asus':7>. The actual *tf* and *idf* values are computed as usual, and normalized by the "document" length. As a result, every token is associated with a weight ranging from 0 to 1.

4. QUERY SEGMENTATION TRAINING

Our goal is to use the automatically labeled query data, obtained as described in Section 3, to train a model to segment queries into attribute-value pairs.

Model: we chose the Conditional Random Field (CRF) model, similar to the one described in reference [1]. Specifically, we train a separate CRF model for each product category (e.g., we train separate models for the "laptops", and "laptop accessories" category), leveraging the fact that every product in our training set is already classified into a distinct category.

Labels: the labels correspond to the set of target attributes, selected as the union of all attribute names matched automatically in Section 3. Following ideas from information extraction [3], we use two types of labels for each attribute, *_begin* and *_continue*. For example, the labels for the sequence of tokens '*hewlett', 'packard*' are *Brand_begin* and *Brand_continue*, respectively.

Features: our segmentation features include Boolean features that represent the presence or absence of token unigrams and bigrams; regular expressions that match different token types (e.g. numbers and words), as well as token context information (e.g. features of the tokens preceding or following the current token).

5. EXPERIMENTAL EVALUATION

The data for the experiments were generated from a sample of the click logs from the Shopping.com product search engine, sampled from the Computers category over a period of approximately one month in late 2009. This included six sub-categories (*Laptops (L)*, *Software (S)*, *Memory (M)*, *Hard Drives (HD)*, *Printers (P)*, and *Laptop Accessories (LA)*). The overall query statistics are in Table 1. A random sample of these queries (approximately 100) were manually labeled to assign appropriate attribute values to each query token if there was one.

Table 1: Combined Query Statistics for All Categories

Queries	*Matched (τ =0.7)*	*Matched (τ =0.3)*
13631	1118	2714

Evaluation metrics: we use the following metrics for our task:

- *Precision:* two variants are used: *Prec(t):* fraction of tokens predicted correctly, and *Prec(q),* computed as *Prec(t)* but micro-averaged across queries.
- *Recall:* two variants are used: *Rec(t)* and *Rec(q)*. *Rec(t)*: fraction of all labeled tokens that were correctly predicted, and *Rec(q):* computed as *Rec(t)*, but micro-averaged across queries.
- *Accuracy*: two variants re used: *Acc(t)* and *Acc(q)*. *Acc(t)*: fraction of correctly predicted tokens, and *Acc(t)*, computed as *Acc(t)*, but micro-averaged across queries.

Table 1 reports the data statistics, i.e., the total number of queries and the resulting number of queries in the training set for similarity thresholds 0.7 and 0.3.

Matching results: first we evaluated the performance of the matching methods in isolation. The accuracy of the similarity method (using a similarity threshold τ of 0.7) is 0.93, compared to the much lower accuracy of 0.43 for the baseline method. However, since there is a trade-off between the matching accuracy and the amount of training data we can generate, we also experimented with lower values of τ. With $\tau = 0.3$, resulting *Acc(t)* was 0.78 and *Acc(q)* was 0.80. The corresponding sizes of the automatically generated training data are reported in Table 1.

Segmentation Results: We use the automatically generated training data from Table 2 with τ=0.3, to train the CRF model with the features described in section 4. The model was evaluated on manually labeled queries that were not included in the training data. The results, averaged over all subcategories, are in Table 2. The accuracy for individual categories ranged from 0.85 to 0.63, with higher performance for more popular (*L, S, M, HD*), and lower for sparse categories with less training data (*P, LA*).

Table 2: Overall Segmentation Accuracy Results (based on manual labels and similarity threshold=0.3)

Method	*Prec (t)*	*Prec(q)*	*Rec(t)*	*Rec(q)*	*Acc(t)*	*Acc (q)*
Baseline	0.229	0.241	0.3	0.17	0.228	0.241
Similarity	**0.75**	**0.748**	**0.753**	**0.757**	**0.736**	**0.738**

6. CONCLUSIONS AND FUTURE WORK

We presented an unsupervised method for query segmentation, based on a robust technique for automatically generating training data. Future research directions include exploring additional methods for generating training data, as well as developing techniques for generalizing the trained segmentation models to other product domains.

7. REFERENCES

[1] Li, X. Li, Y. Wang, and Acero, A. Extracting structured information from user queries with semi-supervised conditional random fields. In Proc. of SIGIR 2009.

[2] E. Agichtein and V. Ganti. Mining Reference tables for automatic Text Segmentation. In Proc. of KDD 2004.

[3] X. Yu and H. Shi. Query Segmentation Using Conditional Random Fields. In Proc. of KEYS 2009.