UNSUPERVISED QUERY SEGMENTATION USING CLICK DATA AND DICTIONARIES INFORMATION

Julia Kiseleva
Saint-Petersburg State University
e-mail: julianakiseleva@gmail.com

Abstract

We describe 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 improving method to automatically acquire such training data — resulting in significantly higher segmentation performance, compared to previously reported methods.

Keywords: *query segmentation, attribute extraction, structured queries.*

1. INTRODUCTION

This work focuses on the problem of detecting and labeling product attribute values in keyword queries to 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 improved compare to [4] *unsupervised* approach to this problem that trains the extraction/segmentation system based on only the product click data. 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 a text segmentation of information extraction system.

This improved unsupervised approach has multiple advantages over the previous supervised and semi-supervised methods [1]:

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

The results, over six categories in the Computer and Electronics domain over real data collected from a leading online comparison shopping site, demonstrate the accuracy and scalability of our approach.

2. RELATED WORK

There are a good number about semi-supervised or unsupervised methods for Conditional Random Fields (CRF) have been published in recent years. This works used additional resources for semi- or un-supervised information extraction.

For example in [2], a database was used to create an artificially-annotated training data to train a language model for Hidden Markov Model (HMM). A similar approach was published in [3] for CRF. Moreover in [5] was proved that adding dictionary as a feature for HMM. This method was shown as effective. In [1] additionally for databases was used clicked data for semi-supervised CRF.

Our previous work [4] was done using only clicked data. In this work we combined this two approaches and used click data and dictionary information for decreasing ambiguity in the result and improving quality. We also extended ground data.

3. SYSTEM OVERVIEW

We developed labeling systems with overall workflow which has two main parts:

- *Part 1*: unsupervised training over automatic labeling.
- *Part 2*: run-time prediction using the trained model and automatically labeled query data.

The system has three main algorithmic components that we optimized:

- Automatic labeling (Section 4);
- Training the segmentation system (Section 5);
- Predicting (applying) segmentation for new queries (Section 6).

After describing the system (Sections 4-5), we introduce our experimental setting including data base, metric and result (Section 6).

4. AUTOMATIC QUERY LABELING

Labeling approach applies to the following problem setup. Shoppers run queries against a product search engine (e.g. "*sony vaio 4gb laptop*", "*sata 200gb*" etc) and clicks on product .These products have a name and textual description, as well as a set of structured attributes represented as name-value pairs (e.g. *name*: "*sony vaio 2.0 ghz intel core duo notebook*"; *set of attributes: 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, i.e. we record clicks on individual products in response to a query, assuming that users tend to click more frequently on products that are highly relevant to the query. Using the obtained query-product pairs, we explored three token labeling methods to generate training data. We explored two methods to generate training data (4.2 and 4.3).

4.1. Tokenization

Because users usually do not care about format of typing query, we should normalize data. The one of issues is tokens related to memory size, hard drive capacity, weight and so on. This issue because one user can write it "2 gb" other "2gb". To avoid ambiguity we tried two approaches:

1. "*Broken tokenization*" — when we put space between pairs like digit and then not digit (we do it as well for database's data). We notice that main trouble with this approach that we broke models like "500d" as well. At the end get "500 d" and it to separate words, and "confusing part" is that "500" could be "*capacity*" as well.

2. "*Join tokenization*" — when we remove space between pairs like digit and not digit. And it works well because now CRF-model obtained mixed token n-gram of this word. This approach improves precision of matching for a few percents and does not hurt recall.

4.2. Baseline Similarity

As a baseline method, we use method proposed in [4]. This method 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. The 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.

Then empirically was found the weight which should show best matched result. By best we mean a reasonable combination between precision and recall. For this we with step 0.1 were counted Precision, Recall for different weight. And it was detected that the best combination of metrics we got with weight 0.3. Therefore a training set supposed to be a collection of queries where every query has weight over that 0.3.

4.3. Similarly

We improve baseline by:

- Identifying synonyms across attributes (section 4.3.1).
- Decrease rarefaction in the data using "updating algorithm" (section 4.3.2).
- Use dictionary for improve accuracy of matching (section 4.3.3).

4.3.1. Synonyms in attributes

We obtain that our set of attributes from matched data. After analyzing we noticed that there are attributes similar to each other. E.g. attributes "*release-year*" and "*release data*" should be equivalent. To detect such kind of synonyms in data base we build dictionary for every attribute in data base and measure how close they are to each other using cosine similarity. Empirically we choose threshold which identifies that 2 attributes is equivalent. This threshold equals 0.7. By introducing synonyms we decreased number of attributes from 65 to 41. It significantly reduces attribute space.

4.3.2. Updating algorithm

In matching algorithms if we can't find any attribute related to particular token we assign "*unknown*" label to it and it has weight equals to zero. Thereafter we notice that the matched data is sparse. By sparse we mean that a sufficient amount of queries contains "*unknown*" token inside. It happens because some description for product could not contain information about some obvious token. E.g. for token "*notebook*" has labeled as "*laptop type*" for significant part of queries but some product description does not contain this token. At the end we obtained a lot of "*unknown*" label and it decrease our training set.

To avoid it we introduce follow procedure:

1. During the matching we collect attributes with set of belonging tokens.
2. For every token calculate a probability to belong to particular attribute.
3. Then do follows:
   - if found "*unknown*" token check collection if it belongs to any of attributes collection;
   - if yes : pick the attributes with highest probability;

As weight for such updated labels we assign a frequency particular label in the collection.

4.3.3. Use dictionary for improve matching

According our experience one of the most important attribute is "*brand*". During the analyzing we obtained such problem as brand could have informal synonyms and we could not identify it from database. For example "*hp*" is synonym for "*hewlett packard*", "*mac*" is synonym for "*mackintosh*" and so on. To avoid it we use brands dictionary with synonyms during the matching.

5. TRAINING SEGMENTATION SYSTEM

5.1. Baseline Segmentation System

As a baseline for segmentation we use following algorithm:

1. 1.Build dictionary from database for every attribute.
2. 2.Calculate posterior probability for every token in this dictionary.
3. 3.Find most likely attribute for every given token in the query.

5.2. CRF Segmentation System

Our goal is to use the automatically labeled query data, obtained as described in Section 4, to train a model to segment queries into attribute-value pairs.

5.2.1. Model

We choose 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.

In the current system we used the Mallet implementation for CRF.

5.2.2. Attributes

The attributes correspond to the set of target attributes, selected as the union of all attribute names matched automatically in Section 4. Then reduce the set of attributes by identifying synonyms in this set of attributes (Section 4.4.1). Following ideas from information extraction, 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.

5.2.3. General 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).

5.2.4. Dictionary Features

To leverage dictionary information to CRF we add follows extra feature:

1. As we discuss before "*brand*" is important attribute. With purpose to identify "*brand*" more easily we add feature which shows is it brand or not. To implement it we use brand dictionary (Section 4.4.1)

2. We created a dictionary for every attribute (Section 5.1) where every attribute is associated with the set of tokens and given token is associated with posterior probability to belong to particular attribute. This information for token we as feature for CRF training.

6. EXPERIMENTAL EVALUATION

The data for the experiments were generated from a sample of the click logs from the 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 Cards (MC)*, *Hard Drives (HD)*, *Printers (P)*, and *Laptop Accessories (LA)*. A random sample of these queries (120 queries from different subcategories) were manually labeled to assign appropriate attribute values to each query token if there was one.

6.1. Evaluation Metrics

We use the following metrics for our task:

Precision: fraction of tokens predicted correctly micro-averaged across queries.

Recall: fraction of all labeled tokens that were correctly predicted micro-averaged across queries.

6.2. Ground Truth

As ground truth we choose 350 unique queries from the log which have length more than 1 token and less than 6 tokens. Compare to [4] where used only 100 queries.

Labeling ground truth was made by qualified Mechanical Turk Users (https://www.mturk.com/mturk/welcome).

6.3. Matching Evaluation

We evaluate matching for different combinations:

1. Use only brand dictionary for improving (DA).
2. Use only "Updating algorithm" (UA).
3. Use DA and UP at the same time.

Table 1. Combined query statistics for matching all categories

Method	Query #	Matched (t=0.7)	Matched (t=0.3)	Time period	Category #
Baseline	13631	1118	2714	3 weeks	6
Brand dictionary		1906	3102		
Updating algorithm		2006	3936		
Overall		**2903**	**4174**		

Table 1 shows that inventing DA and Up together significantly increase training set for prediction model.

Table 2 shows us correlation between increasing training set and improving matching accuracy. We considerably improve recall by adding more data in training set.

Table 2. Overall matching result through all categories

Method	cosine	Precision	Recall
Baseline	0.3	0.80	0.27
	0.7	0.94	0.16
Similarity with Dictionary and Update	0.3	0.84	0.45
	0.7	0.97	0.3

6.4. Segmentation Evaluation

We use the automatically generated training data from Table 1 with τ=0.3, to train the CRF model with the features described in section 5.2.3 and 5.2.4. The model was evaluated on manually labeled queries that were not included in the training data. The results, averaged over all subcategories. The accuracy for individual categories ranged from 0.89 to 0.69, with higher performance for more popular (*L, S, M, HD*), and lower for sparse categories with less training data (*P, LA*).

Table 3 shows results for dictionary approach described in Section 5.1 and results for CRF model trained on set which was described in Section 4 with features in Section 5.2. Compare to our previous result [4] we significantly extend test improve precision. Before ground truth was 100 queries, now — 350 qurties.

Table 3. Overall Segmentation Result trough all categories based on manual labels and similarity threshold=0.3

Method	Precision	Recall
Baseline (Dictionary)	0.55	0.41
Segmentation (CRF)	0.79	0.63

7. CONCLUSIONS AND FUTURE WORK

We presented completely unsupervised method for query segmentation. Furthermore, we present a practical method on how to obtain derived attributes by leverage user click data and domain database in context of tagging commerce search queries. Evaluation shows that our unsupervised method is effective. In future we would like to improve training set quality and extend CRF approach.

8. ACKNOWLEGMENT

We would like to thank the Eugene Agichtein (Emory University), "Shopping.com" and "Ebay" for support.

REFERENCES

1. **Xiao Li, Ye-Yi Wang and Alex Acero**. A Extracting structured information from user queries with semi-supervised Conditional Random Fields. In processing SIGIR, 2009.
2. **S. Canisius and C. Spoleder**. Bootstraping information extraction from the field books. In processing of the 2007 Conference on Empirical Methods in Natural Language Processing, pages 827–836, 2007.
3. **C. Zhao, J. Mahmud, and I. Ramakrishna.** Exploiting structured reference data for unsupervised text segmentation with conditional random fields. In Proceedings of the SIAM International Conference on Data Mining, 2008.
4. **Julia Kiseleva, Eugene Agichtein, Qi Guo, Daniel Billsus, Wei Chai**.Unsupervised Query Segmentation Using Click Data: Preliminary Result. In processing of WWW, 2010.
5. **Willam W. Cohen and Sunita Sarawagi**. Exploiting Dictionaries in Named Entity Extraction: Combining Semi-Markov Extraction Processes and Data Integration Methods. In processing of KDD, 2004.
6. **E. Agichtein and V. Ganti**. Mining Reference tables for automatic Text Segmentation. In Proc. of KDD 2004.
7. **X. Yu and H. Shi**. Query Segmentation Using Conditional Random Fields. In Proc. of KEYS 2009.
8. **T. Crenager, D. Klein, and C.Manning**. Unsupervised learning of field segmentation models for information extraction. In processing of the 43rd Meeting of the ACL, pges 371–378, 2005.
9. **X.Li, Y.-Y. Wang, and Acero**. Learning query intent from regularized click graph. In SIGIR'08: Processing of the 31st Annual International ACM SIGIR conference on Research Development in Informational Retrieval, July 2008.
10. **T.-L. Wong, W. Lam, and T.-S. Wong**. An unsupervised framework for extracting and normalizing product attributes form multiple web sites. In processing of the 31 st Annual International ACM SIGIR conference on Research and development in Informational Retrieval, pages 35–42, 2008.
11. **C. Zhao, J.Mahmud, and I. Ramakrishman**. Exploiting structured reference data for text segmentation with conditional random fields . In Processing of the SIAM International Conference on Data Mining, 2008.
12. **T. Grenager, D. Klein, and C. Manning**. Unsupervised learning of field segmentation models for information extraction. In Processing of the 43th Annual Meeting of ACL, pages 371–378, 2005.
13. **G. Druck, G. Mann, and A. McCallum**. Learning from labeled featutres using generalized expectation criteria. In Processing of the 31st Annual International ACM SIGIR conference on Research and Development in Informational Retrieval, pages 595–602, 2008.