Learning to Tokenize Web Domains

Sriram Srinivasan
Yahoo! India SDC, Torrey Pines,
EGL Business Park, Intermediate Ring Road,
Bangalore - 560071, India
srsriram@yahoo-inc.com

Sourangshu Bhattacharya
Yahoo! Labs India, Torrey Pines,
EGL Business Park, Intermediate Ring Road,
Bangalore - 560071, India
sourangb@yahoo-inc.com

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Learning

General Terms

Algorithms

Keywords

Domain Tokenization, Large Margin Learning, Internet Monetization

1. INTRODUCTION

Domain Match [5] is an Internet monetization product offered by web companies like Yahoo! The product offers display of ads and search results, when a user requests a webpage from a domain which is non-existent or does not have any content. This product receives millions of queries per day and generates significant advertising revenue for Internet companies like Yahoo! Domain Match (DM) works by **tokenizing the input domains** and sub-folders into keywords and then displaying ads and search results queried on the keywords.

In this poster, we describe a machine learning based solution, which automatically learns to tokenize new domains, given a training dataset containing a set of domains and their best tokenizations. For example: `www.marylandregistry.com` => `maryland registry`. Another non-trivial tokenization is: `mary land registry`. We use positional frequency and parts of speech as features for scoring tokens. Tokens are scored combined using various scoring models. We compare two ways of training the models: a simple gain function based training and a large margin training. Experimental results are encouraging.

For training, we use a gain function based on the difference between scores of the best tokenization and other tokenizations. The large margin training is closest to the method used for parsing in [1]. Our method is different from training of CRFs [2], which maximizes the score of the best tokenization. CRFs have been used to tokenize words in Chinese language [4]. This method is not applicable to English language since calculating features describing characters, and deciding to split at a character based on the previous character are not useful. The CRF based method makes more sense when the average length of words is low and features can be attributed to characters. To the best of our knowledge, this is the first machine learning based solution described for English language tokenization.

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0632-4/11/03.

2. PROBLEM DEFINITION

Inputs to the tokenizer consist of a domain, d and a set of terms, F, along with the corresponding features. We use two types of features for scoring terms: search features and dictionary features. Search features are: word (W), start (S), end (E) and intermediate (I); which are counts of occurrences of the term as single word, at the start, end, middle, of a search phrase, respectively in the search logs. Dictionary features are indicators of dictionary word, noun, verb, adjective, adverb. Thus, there are 9 features for each term t, denoted as $a_1(t), \ldots, a_9(t)$ in their order of description.

A term $t \in F$ is scored using a scoring function $S(t)$. We evaluate the following scoring functions:

$$S(t) = l(t) * \log(\mathbf{w}^T \mathbf{a}(t)) \quad w_i \geq 0 \tag{1}$$

$$S(t) = l(t) * \mathbf{w}^T \mathbf{b}(t) \tag{2}$$

where, $b_i = \log(a_i), i = 1, \ldots, 4$ and $\mathbf{w}$ are the coefficients to be learned. For the first model, we only use the search features, and test the second model with and without the dictionary features. $l(t)$ is a linear function of the number of characters in token t.

A tokenization $T(d)$ of domain d is a list of strings $t_1, \ldots, t_n$, called tokens, such that concatenation of t_i's gives d. We choose to score a tokenization by summing the scores of the constituent tokens. Thus $Score(T(d)) = \sum_{t \in T(d)} S(t)$. Let $\tau(d)$ be the set of all possible *valid* tokenizations of the domain d, where at least one token is a term in F. The tokenizer outputs a single best tokenization, by choosing the one that maximizes the score. Thus, final tokenization of a domain d is given by:

$$T^*(d) = \arg \max_{T \in \tau(d)} Score(T) \tag{3}$$

Weighing with the function $l(t)$ ensures that tokenization involving more number of tokens does not get undue preference over the ones involving less number of tokens. The objective is to learn $\mathbf{w}$ from an annotated dataset.

3. LEARNING SOLUTIONS

Let $\mathcal{D} = \{d_i, T_i^* | i = 1, \ldots, N\}$ be the training dataset where d_is are domains and T_i^*s are their best tokenizations. Given a set of coefficient values $\mathbf{w}$ and the dataset $\mathcal{D}$, we can define a gain function $G(\mathbf{w}; \mathcal{D})$ which measures the quality of tokenization.

An intuitive gain function would be to count the number of domains d_i for which the decision function (Eqn 3) selects the best tokenization T_i^* as the optimal one. Thus, gain $G(\mathbf{w};\mathcal{D}) = \sum_{d_i \in \mathcal{D}} \mathbf{1}(T_i^* = \arg\max_{T \in \tau(d_i)} Score(T))$. Unfortunately, this function is very difficult to optimize mathematically. Instead we use the following proxy for gain function:

$$G(\mathbf{w};\mathcal{D}) = \sum_{d_i \in \mathcal{D}} \sum_{T \in \tau(d_i)} (Score(T_i^*) - Score(T)) \quad (4)$$

This gain function is positive for a tokenization T of domain d_i, other than the optimal tokenization T_i^*, whenever T is scored lower than T_i^*. Thus, it tends to make T_i^* highest ranked.

For the first model (Eqn 1), the gain function becomes:

$$G(\mathbf{w},\mathcal{D}) = \sum_{d_i \in \mathcal{D}} \sum_{T \in \tau(d_i)} \left(\sum_{t \in T_i^*} l(t) \log(\mathbf{w}^T \mathbf{a}(t)) - \sum_{t' \in T} l(t') \log(\mathbf{w}^T \mathbf{a}(t')) \right)$$

We maximize $G(\mathbf{w},\mathcal{D})$ w.r.t. $\mathbf{w}$ such that $w_i \geq 0$. Unfortunately, this function is not convex and hence we can only achieve a local optimum. We use a L-BFGS based optimization library [3] to optimize this function. The constraint $w \geq 0$ was enforced using a barrier function. We call this Log-linear model (LLM).

For the second model (Eqn 2), the gain function is:

$$\begin{aligned} G(\mathbf{w},\mathcal{D}) &= \sum_{d_i \in \mathcal{D}} \sum_{T \in \tau(d_i)} \left(\sum_{t \in T_i^*} l(t) \mathbf{w}^T \mathbf{a}(t) - \sum_{t' \in T} l(t') \mathbf{w}^T \mathbf{a}(t') \right) \\ &= \mathbf{w}^T \Delta \end{aligned}$$

where $\Delta = \sum_{d_i \in \mathcal{D}} \sum_{T \in \tau(d_i)} (\sum_{t \in T_i^*} l(t)\mathbf{a}(t) - \sum_{t' \in T} l(t')\mathbf{a}(t'))$. This function is unbounded, since $\mathbf{w}$ can be scaled arbitrarily. Hence, we maximize a regularized version of this function:

$$G'(\mathbf{w}) = \mathbf{w}^T \Delta - \lambda \|\mathbf{w}\|^2$$

We call this the linear model (LM). However, this does not impose a large margin on the correct identification of best tokenization.

Next, we describe a max margin training method for the second model (Eqn 2). For a given domain d_i and a given tokenization $T_j \in \tau(d_i)$, the margin in gain is given as $M_{ij} = \sum_{t \in T_i^*} l(t)\mathbf{w}^T\mathbf{a}(t) - \sum_{t' \in T_j} l(t')\mathbf{w}^T\mathbf{a}(t')$. Let γ be a lower bound on the margin, $M_{ij} \geq \gamma$. Thus, we are interested in maximizing γ. However, this problem is unbounded. Hence, we regularize using the constraint $\|w\| \leq 1$. Thus, the problem can be written as [1]:

$$\mathbf{min_w} \ \|w\|^2; \quad \textbf{subject to:}$$

$$\mathbf{w}^T \left(\sum_{t \in T_i^*} l(t)\mathbf{a}(t) - \sum_{t' \in T_j} l(t')\mathbf{a}(t') \right) \geq 1, \forall T_j \in \tau(d_i), d_i \in \mathcal{D}$$

This problem may become infeasible in many cases. Hence, we add slack variables $\xi_i > 0$ for each domain d_i. Thus, the final formulation becomes:

$$\mathbf{min}_{\ \mathbf{w},\xi} \ \frac{\|w\|^2}{2} + C \sum_i \xi_i$$

$$\textbf{s.t.:} \ \mathbf{w}^T x_{ji} \geq 1 - \xi_i, \forall T_j \in \tau(d_i), d_i \in \mathcal{D}$$

where $x_{ji} = \sum_{t \in T_i^*} l(t)\mathbf{a}(t) - \sum_{t' \in T_j} l(t')\mathbf{a}(t')$. This is a quadratic program which can be solved using any standard QP solver. We call this formulation: large margin linear model (LMLM). [1] use a similar model, except that they use a relative margin with respect to the number of errors made by T_j. In our case, this quantity is not well defined.

4. EXPERIMENTS

We validated our methods on a dataset of 3677 manually tokenized domains from Yahoo! domain match. For generating the set of terms (F), we used a union of Yahoo! search logs and WordNet dictionary. The search logs contained about 1.6M terms, each having W, S, E, and I (first four) features. WordNet, after filtering, gave 77K terms (30K additional) with the dictionary feature set to 1 and the other 4 part of speech features computed appropriately.

We tested 3 models: log-linear (LLM), linear model with gain based training (LM), and linear model with large margin training (LMLM). The linear models were tested with (LM1, LMLM1) and without dictionary features (LM2, LMLM2).

Table 1: 3-fold Crossvalidation Accuracies of various methods for topmost and any of top 3 results

Method	Topmost (%)	Top 3 (%)
LM1	69.35	90.21
LM2	69.16	89.96
LMLM1	70.70	90.58
LMLM2	72.10	91.35
LLM (Untrained)	71.25	91.78
LLM	**78.32**	**94.29**

Table 1 reports the percentage of times the correct tokenization was retrieved as the top ranking result (col. 1) and the top three ranked results (col. 2) by the respective trained tokenizers. Log-linear model performs better than the linear models. Large-margin training improves accuracy of the linear models, for both with and without dictionary features. But the improvement is more in the case where there are more features.

Conclusion. We have looked at the problem of English language tokenization for domains and devised learning algorithms for it. We compare various strategies, and find log linear functions to be more effective than the others.

5. REFERENCES

[1] Michael Collins Daphne Koller Ben Taskar, Dan Klein and Christopher Manning. Max-margin parsing. In *Empirical Methods in Natural Language Processing*, EMNLP 2004, 2004.

[2] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

[3] Jorge Nocedal. liblbfgs: a library of limited-memory bfgs. http://www.chokkan.org/software/liblbfgs/.

[4] Fuchun Peng, Fangfang Feng, and Andrew McCallum. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[5] http://searchmarketing.yahoo.com.