Extracting Advertising Keywords from URL Strings

Santosh Raju
Microsoft Research India,
Bangalore, India
santoshraju@gmail.com

Raghavendra Udupa
Microsoft Research India,
Bangalore, India
raghavu@microsoft.com

ABSTRACT

Extracting advertising keywords from web-pages is important in keyword-based online advertising. Previous works have attempted to extract advertising keywords from the whole content of a web-page. However, in some scenarios, it is necessary to extract keywords from just the URL string itself. In this work, we propose an algorithm for extracting advertising keywords from the URL string alone. Our algorithm has applications in contextual and paid search advertising. We evaluate the effectiveness of our algorithm on publisher URLs and show that it produces very good quality keywords that are comparable with keywords produced by page based extractors.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Keyword Extraction, URL, Contextual Advertising

1. INTRODUCTION

Advertising keyword extraction is important in keyword based online advertising. Traditional keyword extraction methods work on the whole web-page or utilize the anchor text in the inbound links to identify relevant keywords. In this work, we address URL keyword extraction, the problem of extracting keywords from just the URL strings. URL keyword extraction is more difficult than whole-page keyword extraction as there is little context and content available.

URL strings are interesting content available on the web. URLs follow a well defined structure with hostname, category, query and other components. The content in URLs is usually precise and contains text that is highly condensed but relevant to the topic of the web page. Thus URLs can capture user's current interest and are useful in online advertising as a source of relevant keywords. For instance, if a user visits the page `http://blog.syracuse.com/orangebasketball/2010/03/syracuse_vs_georgetown_1.html`, one can infer that the user is interested in a basketball match between Syracuse and Georgetown teams.

There are several advantages to using URL-based keywords in online advertising: (1) URLs are readily available and keywords can be extracted right away without crawling the content of the web page. This reduces the latency in serving advertisements to the new publisher pages. (2) Some publisher pages, especially those which are rich in multimedia content, have very little text content. As a result, only a small number of keywords (or no keywords) might be produced by page-based extractors. In such cases, URL-based keywords are the main keywords for serving relevant advertisements to those pages.

URL text has been used in the past in tasks such as web page classification [1] and URL word breaking [2]. [1] used character n-grams from the URL as features for web page classification and [2, 3] focus on URL and compound word breaking. However, none of these works deal with URL keyword extraction. In this work, we propose URL-KEX, an algorithm for URL keyword extraction. URL-KEX makes use of a keyword dictionary containing an exhaustive set of bid phrases to extract keywords from the URL strings. Further, URL-KEX exploits the structure of the URL to both produce non-trivial keywords and rank the resulting keywords. In the remainder of this paper, we describe URL-KEX and discuss experimental results.

2. URL-KEX

URL-KEX operates in three stages. In the first stage, we extract a first set of keywords from the URL string. In the second stage, we generate more keywords by cleverly combining the keywords from the first stage. In the third stage, we compute a *relevance score* for each keyword and rank the keywords in the decreasing order of relevance.

2.1 Extraction

URL is split into three components based on the structure:*hostname*, *path* and *query*. For instance, `http://local.msn.com/weather.aspx?q=phoenix-az&zip=85004`, is split into *local.msn.com*, */weather.aspx*, *q=phoenix-az&zip=85004*. Further, each component is split into individual units referred to as *segments* based on the delimiters specific to the component. Current example will have the following segments : *local, msn, com, weather, phoenix-az, 85004*[1]. Segments matching well-known noise patterns or strings from a predefined set of stopwords such as "content", "html", "detail" etc are discarded.

Word-Breaking: URL segments often contain compound text strings such as *fraudprevention.* To extract human com-

[1]We retain only values from the key-value pairs in the URL string.

prehensible keywords from URL strings, it is necessary to break the compound text string into series of valid terms. Often, this can be achieved in more than one way. For example, *fraudprevention* can be broken into (fraud, prevention) or (fraud, prevent, ion). While both the splits produce sequences of valid terms, the second sequence is not meaningful. We have observed that when a valid term is split further into valid terms, the resulting split is mostly a meaningless sequence of terms. While for some compound text, it is possible to produce more than one meaningful split, such cases are relatively infrequent and require context beyond the URL string for disambiguation. Our approach is to ensure that all compound text is word-broken while taking care not to split valid terms further.

Our word-breaking algorithm works in two steps: 1) a dynamic program is employed to split the input text into a sequence of valid terms. The dynamic program minimizes the number of terms in the resulting sequence. 2) some of the terms in the sequence produced by the dynamic program might contain compound words which are valid bid phrases but are not commonly used in comparison to some of their splits (e.g. *real estate* vs *realestate*). So every term is further split in all possible ways and the split having the highest bid density[2] is chosen. This will favor *real estate* over *realestate* as the bid density of the former keyword is higher than that of the latter.

Keyword Selection: After word-breaking, keywords are extracted from each segment independently by matching subsequences of terms in the segment with the keywords in the dictionary. The terms are scanned from left to right in the segment and at each term the subsequence of maximum length present in the dictionary is selected. After every match, we jump to the term after the phrase and repeat the same process. A list of common non-informative phrases such as "home page", "about us" is used to filter out some matching subsequences.

2.2 Keyword Synthesis

In the first stage of URL-KEX, keywords are extracted from each segment independently. As each segment of the URL usually represents a distinct aspect of the web page, keywords extracted from the segments could be limited to a single aspect. By combining keywords from different segments, it is possible to produce more keywords as well as more interesting keywords. For example, in `http://local.msn.com/weather.aspx?q=phoenix-az&zip=85004`, the segments *weather* and *phoenix-az* produce *weather* and *phoenix*. By combining the two keywords we get a new keyword, *phoenix weather*, which is more relevant to the web-page than the two constituent keywords. Thus, in the second stage of URL-KEX, we synthesize keywords by combining pairs of keywords from all distinct pairs of segments in the URL and retain only those present in the dictionary and have good bid density.

2.3 Ranking

In the final stage of URL-KEX, keywords from the previous two stages are individually scored and then ranked according to the score. The score of a keyword is based on the following: *parent segment(s) weight* and *keyword length.* The segment weight is based on the component from which it was extracted. Generally, the segments in the *query* component are more informative than those in the *path* and *hostname* components (*query* > *path* > *hostname*). Further, keywords that are informative also tend to be long as they retain a substantial part of the segment. The *score* of a keyword k from a segment s is computed as follows:

$$\text{score} = \frac{s_w k_{len}}{\Sigma_{i=0}^{s_{len}} r^i} \tag{1}$$

where k_{len}, s_{len}, s_w and r are keyword length, segment length, segment weight and decay factor respectively[3]. If the keyword is a combination of two keywords k and k' from two parent segments s and s' respectively, score is computed as follows:

$$\text{score} = \frac{s_w k_{len} + s'_w k'_{len}}{\Sigma_{i=0}^{k_{len}+k'_{len}} r^i} \tag{2}$$

3. EVALUATION

We evaluated the effectiveness of the algorithm on 300 publisher URLs that were sampled from a large pool of publisher pages of a popular ad network and used a keyword dictionary containing 100M bid phrases. For each URL, the top 5 keywords generated by URL-KEX were labeled from the set *precise, approximate, marginal* and *mismatch.* We compared the performance with a page based keyword extractor (Page-KEX) proposed by [4]. Table 1 shows the NDCG@1,3,5 for URL-KEX and Page-KEX on the same set of 300 URLs. The results demonstrate that our algorithm is quite effective and the quality of keywords is comparable favorably to that of a page based extractor.

	NDCG @ 1	NDCG @ 3	NDCG @ 5
URL KEX	0.60	0.47	0.31
Page KEX	0.41	0.29	0.17

Table 1: NDCG @ 1,3,5 for URL based and Page based keyword extractors

We also computed the percentage of URLs for which the two extractors produced at least one and at least two *precise* keyword(s). Page-KEX produced at least one and at least two *precise* keyword(s) for 58% and 37% of the URLs respectively whereas URL KEX was able to do so in only 44% and 26% of the cases. This was clearly due to the limited text available in URL strings. While URL-based extractors cannot fully replace page-based extractors, they produce good quality keywords and are useful.

4. REFERENCES

[1] E. Baykan, M. Henzinger, L. Marian, and I. Weber. Purely url-based topic classification. In *Proceedings of WWW '09.*

[2] S. Khaitan, A. Das, S. Gain, and A. Sampath. Data-driven compound splitting method for english compounds in domain names. In *Proceedings of the CIKM '09.*

[3] K. Wang, C. Thrasher, and B.-J. P. Hsu. Web scale nlp: a case study on url word breaking. In *Proceedings of WWW '11.*

[4] W. Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *Proceedings of WWW '06.*

[2]Bid density of a keyword is the number of ads bidding on that keyword.

[3]In our experiments, $r = 0.5$.