Automatically Annotating the ODP Web Taxonomy

Christiana Christophi† Demetrios Zeinalipour-Yazti†
Marios D. Dikaiakos† Georgios Paliouras‡

† Department of Computer Science
University of Cyprus, CY-1678, Nicosia, Cyprus
{cs98cc1, dzeina, mdd}@cs.ucy.ac.cy

‡ Institute of Informatics and Telecommunications,
National Center for Scientific Research "Demokritos"
Athens, Greece
paliourg@iit.demokritos.gr

Abstract. In this paper we present the ideas and algorithms developed around our *KeyGen* Web Taxonomy Annotation engine. KeyGen annotates the Open Directory Project, also known as Dmoz, with meaningful and previously unknown keywords by utilizing domain knowledge extracted from the WWW. We present two algorithms: i) The *PageParse Algorithm*, which efficiently extracts keywords from Web Taxonomies using a combination of local and global scores, and ii) the *Support Algorithm*, an I/O optimized algorithm for coalescing hierarchies of keywords. We then present the results: i) from constructing a richly annotated ODP Web taxonomy and ii) from evaluating the correctness of this structure by performing an automated classification of Web-pages.

1 Introduction

Thematic taxonomies are tree structures that organize knowledge domains as hierarchies of *concepts* (topics). The most specific concepts are placed at the leaves of these hierarchies while higher-level concepts are assigned to internal tree-nodes. Thematic taxonomies have been used in the establishment of *Web taxonomies* ever since the advent of the WWW (e.g. Yahoo!). Web taxonomies comprise large Web-page repositories whose documents are classified in the various categories (concepts) of selected thematic taxonomies. Web taxonomies are used to enhance the effectiveness of information retrieval on the Web representing an alternative approach to keyword-based searching or hypertext navigation [6]. Web taxonomies have also been used effectively to guide user navigation within large Web directories (portals), to drive focused crawlers, to support categorized searching, document classification [20] and personalization [5, 8, 11]. The construction of Web taxonomies is currently done manually and involves significant effort and cost, although some approaches have also studied automated techniques to do so [19]. The URLs within a given directory might appear either alphabetically (e.g. DMOZ), by the Pagerank of the specific Webpage (e.g. Google Directory) or using a ranking that is deducted through other means [13].

The Open Directory Project (ODP), also known as Dmoz [2], is currently the largest, most comprehensive, human-edited Web Taxonomy. It is constructed and maintained by volunteer editors who insert manually new pages in their category of expertise and annotate these pages with semantic information. ODP classifies over 4 million sites into 590,000 categories with the help of 75,151 editors (circa March 2007). Typical ODP annotations provide short descriptions of the classified pages and associations with other Web-pages. These annotations are provided by the editors based on their personal rules of fairness and objectivity. While these annotations usually capture correctly the meaning of the respective pages, they fail to capture all the important keywords that characterize a given page.

Manual classification of Web-pages into a predefined Web taxonomy structure has been efficient and practical over the years, as this has been demonstrated by the longevity of Yahoo and Dmoz, but annotating these taxonomies is a more challenging problem with no adequate solutions to this day. In particular, current annotations do not allow users to efficiently discriminate in real-time between a large number of concepts or pages that fall under the same category. An erroneous navigation decision can easily divert the user away from the desired resources creating in that way frustration and uncertainty about the effectiveness of the taxonomy.

In this paper we propose an efficient solution to the problem of automatic annotation of large Web taxonomies. Our solution exploits the taxonomy editor's knowledge, which is embedded in the structure of the taxonomy itself and analyzes the page content. Recall that the classification of Web-pages in such taxonomies is traditionally carried out by the editors thus is accurate and relevant. Our ideas are developed in the context of the *KeyGen* system, a high performance system that utilizes the Web-pages referred through the Open Directory Project in order to enrich ODP with meaningful and previously unknown annotations. KeyGen combines powerful information retrieval and indexing techniques to extract keywords from a massive corpus of Web-pages classified in the ODP, and to group keywords into thematic, weighted keyword-sets.

KeyGen initializes its operation by downloading the Open Directory Project (ODP) taxonomy RDF structure [2]. This provides the system with a flat list of m URLs. It then crawls all the respective URLs to a local repository and processes the acquired data in order to construct a lexicon that can be utilized to enrich the ODP taxonomy with new keywords. This process proceeds in the following phases: i) The *Keyword Extraction Phase* using *PageParse* and ii) the *Concept Hierarchy Annotation Phase* using the *Support Algorithm*.

Our Contribution

- We propose and evaluate *PageParse*, which is an efficient and scalable technique for extracting keywords from pages that are referred through Web taxonomies, using a combination of local and global weighting scores.
- We propose the *Support Algorithm*, which efficiently merges keyword sets using a threshold that prunes away keywords with a low support. Our scheme

is designed to optimize I/O performance, as this parameter is of major importance in large repositories of documents.
– We provide experimental results which were obtain from a real ODP Web taxonomy snapshot, which consists of more than 4.5 million keywords.

2 Related Work

In this section we provide an overview of related research work. Although the problem of extracting semantically correlated keywords from a corpus of documents or the WWW, as well the problem of classifying Web-Pages into a pre-constructed taxonomy, are both well studied, exploiting large human-edited Web taxonomies in order to achieve both tasks has not been studied in any other context.

Document Classification in Thematic Taxonomies
The automatic annotation of Web Taxonomies studied in this paper is quite different from the typical supervised or unsupervised *classification problem* in the context of Web Taxonomies [4, 12, 23]. In the classification problem, we start out from an empty taxonomy of predefined concepts T and some unclassified document u, in order to assign u to the appropriate topic in T. In our context, T is already partially or fully constructed by editors, who manually insert URLs in the respective categories, thus there is no requirement to *bootstrap* [4] the taxonomy. The challenge in our context is to automatically annotate the taxonomy T with new keyword annotations, in such a manner that T can be exploited by taxonomy users for better navigation, by focused crawlers to contextually prioritize their crawling sequence and by other applications that take a Web taxonomy as an input.

Semantically-Correlated Keyword-Sets
Constructing a keyword-enriched taxonomy creates in essence a hierarchy of semantically correlated keyword-sets. There are many tools for creating such semantically correlated keyword-sets [3, 25], but none of these takes into account the hierarchical relation between these keywords. For instance, Google Sets [1], a research tool that allows users to automatically create sets of items from a few examples, expands a small number of representative candidates from a given concept given by the users. By providing terms, such as *bmw,vw,audi*, the system lists all available car manufacturers. However this service only considers a flat structure of keywords rather than a hierarchy. This *Query By Example* paradigm has been extensively used in content-based image retrieval (e.g. the QBIC [10] project at IBM Almaden), content-based audio retrieval [22] as well as SQL databases (Microsoft Access) and is appropriate when a query-set is available.

Web Taxonomies
The Open Directory Project and Web Taxonomies in general, have been utilized by a number of other projects, because they organize the WWW into a hierarchy of high-quality recommendations. In particular, Web taxonomies are used to

enhance the effectiveness of information retrieval on the Web, representing an alternative approach to keyword-based searching or hypertext navigation [6]. Web taxonomies have also been used effectively to guide user navigation within large Web directories (portals), to drive focused crawlers, and to support categorized searching and document classification [4, 5, 8, 11, 21, 18]. The ODP taxonomy has been utilized in the context of personalized Web search [14, 9, 15] where it refines query answers that are returned through general purpose search engines. The ODP structure has also been utilized by Chakrabarti et al. in [7], in order to study the topic properties of the Web.

3 PageParse: The Keyword Extraction Algorithm

In this section we describe the intuition behind the PageParse keyword extraction algorithm, the module that extracts keywords from pages referred through the ODP taxonomy. Note that the ODP directory has already been downloaded to local storage by our Web Crawler at this point.

3.1 Pre-Processing Phase

Let W = $\{w_1, w_2, ..., w_m\}$ denote our collection of m Web pages. In the first phase of PageParse, we extract all the keywords from the collection W and create a set of r keywords $K = \{k_1, k_2, ...k_r\}$. Recall that Web-pages in our Information Repository are documents encoded in HTML. HTML represents a primitive structure of each document through the use of tags that specify document features such as the title, the main body, various headings, characteristic keywords, and anchors. Intuitively, terms that appear in the title, header, keyword meta-tags or have special typesetting demarcation (bold, emphasis, etc.) can be considered to be semantically more important than plain document text for the characterization of the document's contents. For this reason, KeyGen exploits the Web-page structure by assigning different importance weights to terms according to the HTML tag that they are assigned (see Table 1).

Table 1. *HTML Tag Classes*: KeyGen calculates the frequency of appearance of a term in each of the seven classes of HTML and produces a *Term Frequency Vector (TFV)*

	Class	**HTML Tags**
1	Title	$\langle TITLE \rangle$
2	Meta Data	$\langle META \rangle$
3	Headings 1,2	$\langle H1 \rangle, \langle H2 \rangle$
4	Headings 3-6	$\langle H1 \rangle - \langle H6 \rangle$
5	Anchor	$\langle A \rangle$
6	Emphasized	$\langle B \rangle \langle I \rangle \langle STRONG \rangle \langle EM \rangle \langle DL \rangle \langle OL \rangle \langle UL \rangle$
7	Plain Text	Everything else

In order to avoid commonly appearing words, our system utilizes a stop-word list of keywords, eliminating in that way frequently-used words, such as "and," "yes," "me," "do," "take", which do not provide any semantic benefit to our lexicon. We furthermore apply the widely used Porter stemming algorithm [16], in order to reduce common morphological and inflectional endings from words in our list of keywords K.

3.2 Weighting Scheme

A document (or downloaded URL) in our setting, is represented as a vector in a r-dimensional Euclidean space, where r is the size of all the unique keywords in the repository. The coordinate (weight) of each term in each document in this space is defined as the product of three parameters depicted in Equation 1:

$$w_{ij} = L_{ij} \times G_i \times N_j \tag{1}$$

1. **Local Weight (L_{ij})**: The Term Frequency Element L_{ij} of word i in document j, which denotes the significance of a term in a particular document;
2. **Global Weight (G_i)**: The Collection Frequency Element of word i, which denotes the importance of the term in the whole Information Repository, and
3. **Normalization Factor (N_j)**: The Length Normalization Element of document j, which is used to avoid the bias of longer documents over shorter ones.

Many different techniques have been proposed for calculating each of these factors. The modularity of KeyGen enables the easy integration of different methods. Currently, KeyGen implements the *Term Frequency (TF)* as a local weight, the *Inverted Document Frequency (IDF)* as a global weight and *Cosine (COSN)* as a normalization length. Therefore equation 1 will be transformed to equation 2.

$$w_{ij} = TF_{ij} \times IDF_i \times COSN_j \tag{2}$$

Term Frequency (TF_{ij}) is the frequency of occurrence of term i in document j. Inverse Document Frequency (IDF_i) suggests that a good term exhibits low collection frequency. Cosine Normalization ($COSN_j$) is a popular normalization factor; it normalizes the weighted document vector so that the magnitude of the weights is one.

4 The Support Algorithm: Merging the Keywords

In this section we present the *Support Algorithm*, which recursively coalesces the keyword-sets identified during the PageParse phase into a keyword-enriched concept hierarchy $T = \{t_1, t_2, ..., t_m\}$. Since we have to cope with a very large collection of concepts and Web-pages stored on secondary storage, we seek to optimize I/O access. In the experimental section, we will show that our collection consists of over 1,000,000 Web-pages classified in more than 78,000 concepts. Our algorithm has a linear time complexity of $O(n + m)$, where n is the number of Web-pages $W = \{w_1, w_2, ..., w_n\}$ and m is the number of concepts $C = \{c_1, c_2, ..., c_m\}$.

4.1 Support Algorithm Description

The Support Algorithm is a recursive algorithm that is highly optimized for I/O efficiency. In particular, our algorithm merges the keyword-sets, available for each Web-page, using a single linear scan over the respective local repository.

Our algorithm performs a Depth-First-Traversal of the tree taxonomy T. At each leaf level, it executes the PageParse algorithm on all the Web-pages referenced from within the given level (See Algorithm 1, line 3-9). This produces a set of keywords per Web-page (*keyset*). The algorithm then hashes all the keywords located in *keyset*, into an in-memory hash table (*pagehash*) in order to enable more efficient joins between keyword-sets in the subsequent steps. Each bucket of *pagehash* maintains both the keyword and the weight of each respective keyword. The weight will be utilized in order to remove keywords that have a weight (*support*) below a given threshold.

Algorithm 1 : Support_Algorithm

Input: n Web-pages $W = \{w_1, w_2, ..., w_n\}$, m concepts organized in a concept hierarchy $C = \{c_1, c_2, ..., c_m\}$.
Output: A keyword-enriched concept hierarchy $T = \{t_1, t_2, ..., t_m\}$. Each t_i consists of a keyword-set that annotates concept c_i.

```
1: procedure SUPPORT_ALGORITHM(C, W)
2:     levelhash{} = 0;                    ▷ Hashtable with keywords identified for this level.
3:     if (isLeaf(C)) then
4:         ▷ Assign the keywords from all Web-pages into levelhash.
5:         for j = 1 to |C| do             ▷ |C|: Number of Web-pages in concept C.
6:             keyset=PageParse(w_j)
7:             pagehash=hash(keyset)       ▷ hash keywords in keyset.
8:             merge(levelhash, pagehash)
9:         end for
10:    else
11:        ▷ Join the keywords from all children concepts into levelhash.
12:        for j = 1 to |C| do             ▷ |C|: Number of Concepts in concept C.
13:            merge(levelhash, SUPPORT_ALGORITHM(c_j, childpages_j(W)))
14:        end for
15:        ▷ Sort in descending order the levelhash based on the weight of keywords.
16:        levelrank = sort(levelhash);
17:        t_current = {}; j = 1
18:        ▷ Filter out keywords that have a weight below τ.
19:        while (levelrank_j.weight >= τ) do
20:            copy(levelrank_j, t_current)
21:            j++
22:        end while
23:        ▷ The keyset for this level are the keywords in t_current.
24:        levelhash = t_current
25:    end if
26:    return levelhash;
27: end procedure
```

While obtaining the appropriate keywords from a given Web-page, *pagehash* is merged with the rest of the keywords that are located at the current level of the taxonomy (*levelhash*). In order to achieve this task, we have to merge two hashtables of size $|levelhash|$ and $|pagehash|$ respectively. This is a fairly cheap task that can be achieved in linear time $O(|pagehash|)$, since each insertion in *levelhash* is performed in $O(1)$. If we chose to implement this procedure without hashtables, we would need $O(|pagehash| * |levelhash|)$ as each of the $|pagehash|$ insertions would require $O(|levelhash|)$. In general, given K keysets, each with an average size of l, would require $O(l^K)$ time in the absence of the hashtables, while our approach requires only $O(l * K)$ time for the hashing and $O(l * K)$ time for the merging (i.e. again $O(l * K)$). Finally note that both *pagehash* and *levelhash* are already in memory, thus this function is extremely fast.

After the *levelhash* tables are created at the leaf levels of the tree T, we proceed with sorting of these hashtables in order to identify the keywords above a certain threshold. This is illustrated in lines 11 to 25 of Algorithm 1. The threshold τ, is a user-defined parameter which identifies the minimum weight a keyword should have in order to be retained for the given level of the taxonomy.

5 Experimental Evaluation

In this section we present our datasets, evaluation parameters and evaluation results. Our system configuration includes a non-exclusive Solaris machine with 4 CPUs and 8GB RAM.

5.1 Description of Datasets

Our dataset consists of the first 6 levels of the ODP Web taxonomy along with their respective Web-pages. In order to obtain this dataset we started out by downloading the ODP directory dump in RDF-XML [1]. We then performed a breadth first traversal for 6 levels of the ODP structure and generated a seed list of 1,153,086 URLs (coined the *ODP seed list*). The seed list was provided as an input to the open source WebRace crawler [26], which downloaded the respective pages into our local repository. Since many of the given URLs were either not available at the time of our crawl, or had been completely eliminated from the respective Web servers, we were only able to download the 91% of the complete list (i.e. 1,046,021 URLs). The downloaded Web-pages required about 4GB of hard drive in a compressed form. Since our intention was to build a multilingual corpus of semantically correlated keywords for each given topic, we did not limit our crawl to the content of a specific language but rather utilized all of them. We note that 257,978 pages were listed under the top-level concept *Top/World*, which are pages classified as non-English.

[1] http://rdf.dmoz.org/

5.2 Generating the Lexicon

We processed the first 6 levels of the ODP Web taxonomy, which were previously stored to local storage, using the PageParse algorithm. This resulted in a collection of 4,634,247 unique (and stemmed) keywords. These keywords represent 78,312 unique topics. By excluding the Top/World branch of the ODP structure, which includes mainly non-English content, we derived a sub-lexicon of 3,468,071 keywords and 65,165 topics. This shows that our system is able to uncover an extremely large number of keywords and topics.

By analyzing our acquired data we found that each topic features at least one keyword, at most 300K keywords and on average 1,689 keywords. Although each topic usually contains several keywords, many of these keywords have a low weight. These keywords will be excluded using the user defined pruning threshold τ and will not be utilized to characterize a given topic. In an attempt to limit the words to English ones, we removed the Web-pages that belonged to topics under *Top/World*. The majority of these pages are multi-lingual. In the graph in Figure 1 it is illustrated how the number increases rapidly when pages that include non-English pages are processed, as opposed to the case where non-English pages are removed.

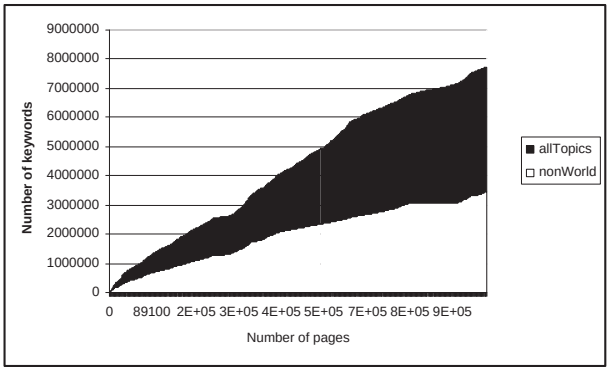

Fig. 1. Increase of number of keywords as the number of Web-pages increases

5.3 Evaluating Precision through Web Page Classification

In the second experimental series, we perform a classification of Web-Pages using the keyword-annotated ODP structure constructed during the previous process similarly to [20]. The intuition behind these experiments is that the improved keyword-annotated topic keywords should be able to more accurately classify a given URL to some topic. Note that the classification of URLs into topics, in a real setting, is a manual process which is conducted by editors based on their domain expertise. On the other hand, performing this action automatically can

assist editors in this non-trivial process. Note that the ODP structure classifies 4 Million URLs, while search engines, such as Google, index more than 8 Billion documents. Thus, the scalability of manual classification systems is extremely limited, and we would like to offer automated means for scaling this process.

In order to evaluate the accuracy of the automatic classification of URLs into topics, we adopt the following methodology: During the construction of the keyword-annotated ODP structure, we excluded a set of URLs. Since these URLs belong to the ODP structure, we precisely know their topic mapping. We then classify these URLs using the WEKA [24] classification algorithms and observe the percentage of accurately classified instances. As a measure of accuracy, we take the ratio of correctly classified URLs versus all classifications:

$$Accuracy = \frac{\#\text{ of correctly classified URLs}}{\#\text{ of classified URLs}} \tag{3}$$

In our experiment, we limit our dataset to documents that belong to topics of depth 2 in the ODP taxonomy. The dataset consisted of 3,940 documents, 194 topics and 16,373 words. Each of the 3,940 instances contained the weights of the 16,373 keywords in the particular document, as well as the real topic to which the document belongs. By classifying these pages using the Nearest Neighbor algorithm, with 5 neighbors, yielded an accuracy of 85.42% of correctly classified instances. This shows that by utilizing the annotated ODP directory might be useful in automatically classifying new web pages to the directory.

6 Conclusions and Future Work

The focus of this paper is to produce a good-quality, large-scale database of keywords for the ODP topic taxonomy. We have proposed KeyGen, a scalable, flexible, and highly customizable system that can process a massive corpus of Web documents. A keyword-enriched taxonomy has a wide array of interesting applications, such as Focused Crawling [8], Pay-Per-Click Advertising [17] and Automatic Classification of Web-Pages [4, 12, 23]. In the future we plan to make our system and datasets open source. We additionally plan to apply the results of this paper in one or more of the aforementioned applications. One final direction is the optimization of the components that comprise the KeyGen architecture.

References

1. Google sets. http://labs.google.com/sets.
2. Odp - the open directory project: Available online at: http://dmoz.org/.
3. Releword, relekey, reletext: http://www.relevad.com/.
4. G. Adami, P. Avesani, and D. Sona. Clustering documents into a web directory for bootstrapping a supervised classification. In DKE, 54(3):301–325, 2005.
5. C. C. Aggarwal, F. Al-Garawi, and P. S. Yu. Intelligent Crawling on the World Wide Web with Arbitrary Predicates. In WWW'01.
6. S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data.* Morgan Kaufmann Publishers, 2003.

7. S. Chakrabarti, M. M. Joshi, K. Punera, and D. M. Pennock. The structure of broad topics on the web. In WWW '02.
8. S. Chakrabarti, B. M.V.D, and B. Dom. *Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery.* In WWW'99.
9. P. Chirita, W. Nejdl, R. Paiu, and C. Kohlschuetter. Using ODP Metadata to Personalize Search. In SIGIR'2005.
10. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The qbic system. *IEEE Computing Magazine*, pages 23–32, 1995.
11. M. A. Hearst and C. Karadi. Cat-a-Cone: an interactive interface for specifying searches and viewing retrieval results using a large category hierarchy. SIGIR '97.
12. D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In ICML '97.
13. V. Krikos, S. Stamou, P. Kokosis, A. Ntoulas and D. Christodoulakis, DirectoryRank: ordering pages in web directories. In WIDM'05.
14. S. Middleton, D. D. Roure, and N. Shadbolt. Capturing Knowledge of User Preferences: ontologies on recommender systems. In ICKC'01.
15. S. Oyama, K. Tanaka, T. F., and M. L. Persona: A contextualized and personalized Web search. In HICSS'02.
16. M. F. Porter. *Readings in Information Retrieval*, chapter An algorithm for suffix stripping. Morgan Kaufman, San Francisco, 1997.
17. B. Ribeiro-Neto, M. Cristo, P. B. Golgher, and E. S. de Moura. Impedance coupling in content-targeted advertising. In SIGIR '05.
18. K. Stamatakis, V. Karkaletsis, G. Paliouras, J. Horlock, C. Grover, J. R. Curran, and S. Dingare. Domain-specific Web site Identification: The CROSSMARC Focused Web Crawler. In WDA 2003.
19. S. Stamou, V. Krikos, P. Kokosis, A. Ntoulas, and D. Christodoulakis, Web Directory Construction using Lexical Chains. In NLDB'05.
20. S. Stamou, A. Ntoulas, V. Krikos, P. Kokosis, D. Christodoulakis. Classifying Web Data in Directory Structures. In APWEB'06.
21. S. Sizov, M. Theobald, S. Siersdorfer, G. Weikum, J. Graupmann, M. Biwer, and P. Zimmer. The BINGO! System for Information Portal Generation and Expert Web Search. In CIDR'03.
22. A. Wang. The shazam music recognition service. *Commun. ACM*, 49(8):44–48, 2006.
23. K. Wang, S. Zhou, and S. C. Liew. Building hierarchical classifiers using class proximity. In VLDB'99.
24. I. H. Witten and E. Frank. *Data mining: practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann Publishers Inc., 2000.
25. W. Yih, J. Goodman, and V. Carvalho. Finding advertising keywords on web pages. In WWW '06.
26. D. Zeinalipour-Yazti and M. Dikaiakos. Design and Implementation of a Distributed Crawler and Filtering Processor. In NGITS'2002.