An Efficient and Effective Algorithm for Hierarchical Classification of Search Results

Masayu Leylia Khodra[1*], Dwi Hendratmo Widyantoro[1]

[1] *School of Electrical Engineering and Informatics – Institut Teknologi Bandung,Jl. Ganesha 10 Bandung*

This paper presents an efficient yet effective algorithm to hierarchically organize search results. Rather than using clustering technique, this paper employs domain ontology in order to obtain better hierarchical classification. Domain ontology defines information architecture in a specific domain. The hierarchical classification process consists of two stages. First, in off-line mode, a classifier is employed to determine category in ontology that is similar to a Webpage. Second, when processing a user's search query, all search results are hierarchically categorized using the classification scheme provided in the metadata of retrieved documents.

1. Introduction

As the volume of information available on the internet continues to increase, internet will become the main information sources. By using search engine, users fulfill their information needs. Most of search engines present list of search results linearly based on search queries. A user must examine each document in the list before the relevant information is found.

Organizing search results into hierarchical structure can help users navigate, seek and find more quickly information they are looking for. This paper presents an efficient yet effective algorithm to hierarchically organize search results. Although efficiency and effectiveness are two contradictory performance measures (i.e., hard to achieve simultaneously), we tackle this difficulty by analyzing the problem domain, and trading-off design choices in order to maximize the desired outcome. Existing search engines group search results (if they do) by using clustering techniques. Because the clustering process must be performed quickly, it significantly reduces the cluster quality. Even when given more time to cluster the search results, the cluster quality is still far from satisfying and is often confusing because search results can be clustered in so many ways.

Rather than using clustering technique, this paper employs ontology in order to obtain better hierarchical classification. Ontology defines information architecture in a specific domain. It contains concepts about information classes (categories) and their hierarchical relationships. Since ontology is a domain specific, this approach is not suitable for general purpose (domain-free) search engine (in separate paper we propose a new search engine architecture that addresses this problem).

The rest of this paper is organized as follows. Section 2 discusses related works. The main concepts are provided in Section 3. The development of hierarchical classification algorithm is described in Section 4. The last section gives concluding remarks and future work.

2. Related Work

There are three approaches in organizing search results:

1. Manual classification for each web site to a proper category in the directory structure, for example, Yahoo! Directory (1) and other similar Internet Directories. This approach results a high-quality search results classification. Because it involves human effort for classifying each Web Site, this approach in general is not scalable, expensive to build and maintain.
2. Automatic clustering search results to hierarchically organize search results. This approach was used to group the search results by Hearst and Pederson (Scatter-Gather clustering algorithm (2)), Zamir and Etzioni (suffix tree clustering (3,4) for HuskySearch meta-search engine), and Cheng et al. (divide and merge clustering (5)). Vivisimo is an example of commercial search engine that employ incremental clustering to do similar task (6). The approach is scalable and can be applied to general-purpose search engine. However, this approach also has a drawback. Because there are so many ways to cluster search results and the facts that the data to be clustered (i.e., text) are naturally noisy, the quality of clustering results is often low and difficult to interpret.
3. Ontology-based search result classification. This research uses this semi-automatic approach, trading-off between fully automatic mode (through clustering) and manual mode (such as through Internet Directory). Ontology, which defines the information structures and classes, is built manually in order to get high-quality information architecture. The classification process that organizes the search results is performed automatically.

3. Main Concepts

The main concepts that are related with algorithm development in this paper are including ontology and vector space model.

3.1. Ontology

Ontology is a specification of conceptualization (7), that is description of concepts and their relationships. Ontology is composed of concepts, attributes and the relation among concepts. Concept is anything that can be described. It can be a real, fictive, concrete or abstract. A concept in ontology can be described by the assignment value of its attributes.

Ontology is at the heart of this research, which defines the concepts about Web page (document) categories and their hierarchical relationships. The main function of ontology is to provide the knowledge base needed for the classification of search results. Because the definition of each information class is crafted manually by a human expert, it is expected that the classification accuracy will be better than those produced by clustering techniques.

* Email: masayu@informatika.org

Fig. 1 provides example of university ontology describing the concept hierarchy in university domain. Pendidikan (education), Penelitian (research), Pengabdian (public service), Kemahasiswaan (student), and Fasilitas (facility) are sub-concepts of university. Furthermore, leaf node concepts also have attribute values representing the characteristics of the corresponding concepts. In this example, komunitas (community), and masyarakat (people) are terms representing the concept of Pengabdian (public service).

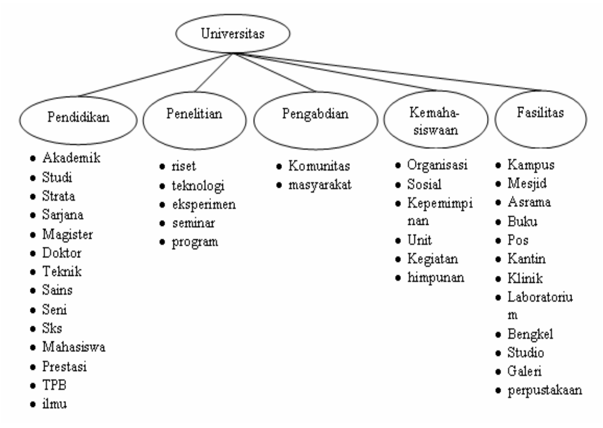


Fig. 1. Example of university ontology

3.2. Vector Space Model

Vector Space Model (VSM) is general model for representing text documents. In this research, VSM is used not only for document representation, but also for concepts definition.

Text documents in VSM are defined as n-dimensional feature vectors. Let D be a text document, then $\{(t_1, w_1), (t_2, w_2), \ldots (t_n, w_n)\}$ is a feature vector of D where t is a *term* occurring in D and w is the weight of *term t*. This representation requires a method to weight each term and to measure the similarity between two document feature vectors.

Ontology concepts in VSM are also represented by n-dimensional feature vectors. Every leaf concept has list of term that manually defined by human-expert when defining the ontology or automatically resulted from predefined document collection. Weight of each term is determined by average frequency in documents with the same category.

Term Weighting. Term weighting is concerned with the assignment of a value to a term that represents the degree of importance of the term in a document. For example, a term that is more important will be assigned a higher value than a less important one. This research used feature vector of Lucene including it's term weighting.

Similarity Measure. A document similarity measure assesses the degree to which a document matches a reference feature vector. This metric is usually used to evaluate documents in order to rank them and then filter those that are not relevant to the information need. In the vector space model, the *cosine* formula is the most widely used similarity measure (8).

This similarity measure calculates the difference in direction between two feature vectors, which is basically the angle between these feature vectors, irrespective of their length. Given a web page $\boldsymbol{D}_i$ and set of concepts descriptions (models of information class) that are leaf nodes in the ontology, the classification of web page $\boldsymbol{D}_i$ according to cosine formula is given by

$$v_{vsm} = \arg\max_{v_j \in V} Sim(D_i, v_j) \tag{1}$$

Hence, the topic of a web page is determined by information class defined in ontology whose similarity to the web page is the closest.

To avoid misclassifying a web page that is totally dissimilar to any concept in ontology, a threshold can be applied to make a final decision. In particular, if the maximum similarity is lower than a specified threshold value, the web page topic in question is declared to be unknown.

4. Development of Hierarchical Classification Algorithm

Fig. 2 shows the hierarchical classification process that consists of two stages: offline stage, and online stage.

Offline stage encodes classification scheme metadata for each web page. We use Heritrix (9) as web crawler to collect web page in itb.ac.id domain.

In online stage, all search results are hierarchically categorized using the classification scheme provided in the metadata of retrieved documents. Classification scheme is a total ordering class from the most general (i.e. root of ontology) to the most specific class (i.e. leaf of ontology). We use Lucene (10) as search engine. We combined Lucene with interactive navigation interface generator, that uses this hierarchical structure to present list of search results hierarchically.

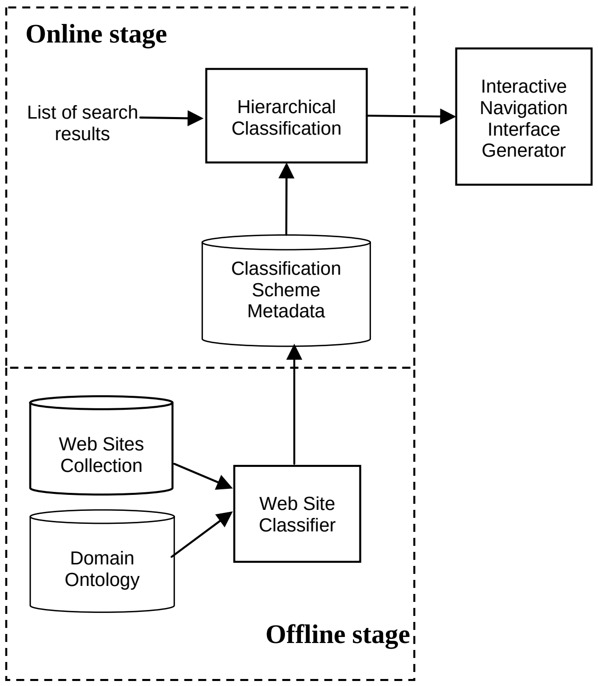


Fig. 2. Hierarchical Classification Process

4.1. Stage 1 – Offline Stage

Web page classification process in this research is conducted off-line, like the process of creating inverted file index of search engine, in order to help speed up the hierarchical classification process. A vector space model based classifier is employed to determine category in ontology that is similar to a Webpage by using equation 1. Classification process results metadata collection. This metadata contains information about the Web page identification, and webpage classification scheme which is a total ordering class from the most general (i.e. root of ontology) to the most specific class (i.e. leaf of ontology).

$$\texttt{<WebPage\_ID, <[Concept\_Name]}^{+}\texttt{>>}$$

As an example, suppose a web page `d1` belongs to penelitian concept (see Fig. 1). Then the classification schema metadata for `d1` is as follows:

```
<d1, <universitas, penelitian>>
```

Fig. 3 summarizes the process of off-line classification of Web site collection and the creation of classification-scheme metadata. Each Webpage is annotated with a metadata that encodes the webpage classification scheme. In implementation, each metadata was saved in Lucene's index as a field for every document.

```
S ← a set of leaf node concepts in ontology
D ← web site collection

For each $d \in D$
  // Find the most similar concept S to d
  $c = \arg\max_{s \in S} Sim(s,d)$
  If Sim(c,d) ≥ θ then
      // create metadata for d
      metadata ← <ancestors(c), c>
      save metadata to classification scheme
      metadata collection
```

Fig. 3. Web site classification process

4.2. Stage 2 – Online Stage

The second stage is performed when processing a user's search query. All search results are hierarchically categorized using the classification scheme provided in the metadata of retrieved documents.

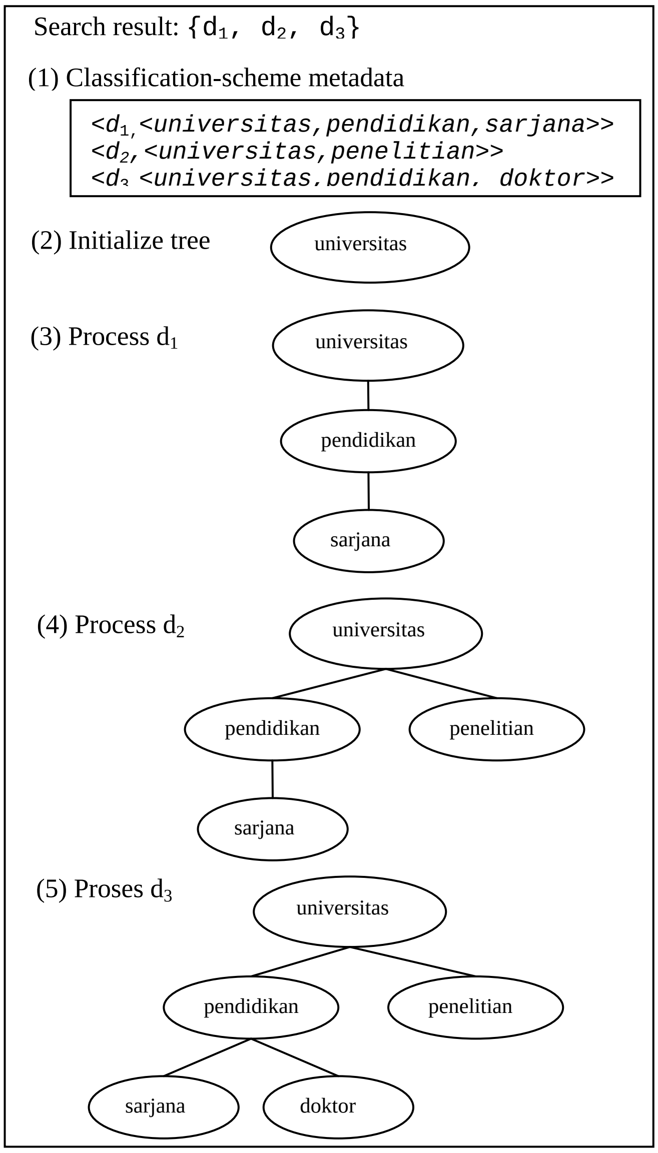
Each metadata has association with a concept that is a leaf node of domain ontology. We use ontology structure to arrange hierarchical structure. Fig. 4 summarizes the process of hierarchical categorization.

Let D be a set of web pages returned by conventional search engine, the hierarchical classification process is performed as follows. First, retrieve the classification scheme metadata for each web page in D. Second, initialize the tree with ontology root. Finally, build a tree structure based on the classification scheme defined in each metadata.

```
C ← a set of classification-scheme metadata
T ← <root,nil> //format <root, list_of_child>
For each metadata ∈ C
    //skip first concept (root)
    metadata ← metadata – first concept
    addLabel (T,metadata)

Procedure addLabel(tree, metadata)
    if metadata ≠ null //metadata has concept
        //take first concept
        c1 ← first concept of metadata
        If c1 ∉ child_tree
            tree.addChild(c1)
            tree ← c1
        else //c1 ∈ child_tree
            tree ← tree.getChild(c1)
        metadata ← metadata – c1
        addLabel(tree,metadata)
```

Fig. 4. Hierarchical categorization process



```
<d1,<universitas,pendidikan,sarjana>>
<d2,<universitas,penelitian>>
<d3,<universitas,pendidikan, doktor>>
```

Fig. 5. Web site classification process

Fig. 5 illustrates the hierarchical classification described above. Supposed the search engine returns three web pages and the associated metadata of these pages are showed in step 1. After tree initialization in step 2, the construction of tree based on the classification scheme on these metadata results from step 3 to 5.

5. Concluding Remarks and Future Work

Development of this hierarchical classification algorithm is part of research in development of next generation search engine. Although there are two stages for this algorithm, the hierarchical classification process in online stage can be performed very fast because of the classification scheme encoded in the metadata.

Future work of this research will be focused on the integration of web page classification and indexing processes in offline stage, integration of hierarchical categorization and hierarchical clustering (for documents that have unknown classifications because maximum similarity is lower than threshold), and design of better university ontology.

References

(1) http://www.yahoo.com
(2) M. Hearst, J. Pedersen & D. Karger, "Scatter/gather as a tool for the analysis of retrieval results", Working Notes of the AAAI Fall Symposium on AI Applications in Knowledge Navigation (Cambridge MA, November 1995).
(3) O. Zamir & O. Etzioni, "Grouper: A dynamic clustering interface to web search results", Proceedings of WWW8 (Toronto, Canada, May 1999).
(4) O. Zamir & O. Etzioni, "Web Document Clustering: A Feasibility Demonstration", ACM SIGIR'98.
(5) D. Cheng, R. Kannan, S. Vempala, G. Wang (2005), A Divide-and-Merge Methodology for Clustering, ACM PODS 2005
(6) http://www.vivisimo.com
(7) T.R. Gruber. A Translation Approach to Portable Ontologies. Knowledge Acquisition, 5(2):199-220, 1993.
(8) I.Witten, A.Moffat, , & T.C. Bell, (1994) Managing Gigabytes: Compressing and Indexing Documents and Images. New York: Van Nostrand Reinhold.
(9) Heritrix project homepage, http://crawler.archive.org
(10) Lucene search engine, http://lucene.apache.org