Organising Web Documents into Thematic Subsets using an Ontology (THESUS)

B. Nguyen[1], M. Vazirgianis[1,2], I. Varlamis[2], M. Halkidi[2]

[1]INRIA
Domaine de Voluceau
78152 Le Chesnay CEDEX
FRANCE
Firstname.Lastname@inria.fr

[2]Athens University of Economics and Business
76 Patision Street
Athens, 10434
GREECE
{mvazirg, varlamis, mhalk}@aueb.gr

Abstract
We describe in this article the architecture and of a system whose goal is to organise Web documents into clusters. We use incoming links to find accurate keywords describing each document. We assume an ontology of the domain we are interested in, and we use a thesaurus (WordNet) in order to map the keywords that describe a document to terms of the ontology. We then cluster the documents using a novel similarity measure, and a modified version of the incremental DB-Scan algorithm.

A. Introduction

The general context is that of a user looking to construct a data warehouse on a specific domain, by using the information found on the Web. For a complete outlook on the problem, we refer the reader to [HNV+02], in the following paragraph we detail the core ideas of the system and the assumptions we hold for this article.

We suppose that this user is a “specialist” of the domain, and has at his disposal (or has constructed) an *ontology* (IS-A tree) describing the domain. In [HNV+02] we propose the architecture of a system that lets a user construct a data warehouse on the topic described by this ontology, using the following modules : page fetcher, semantics enhancer, classifier, query engine. In this article, we detail the system architecture. It is important to note that the input of the classifier is a set of documents, that have attached to them a set of weighted *terms* of the domain ontology. We base most of our reasoning in the article on the assumption that **we are provided with such a set of terms**. For more details on how this set is constructed, we refer to [VN+02] and [HNV+02]. The goal of our system is the following: *Given documents that are characterized by a (short) set of weighted terms from an ontology, find a way of clustering related documents together*. We propose a clustering scheme, based on a novel similarity measure between sets of terms that are hierarchically related.

Traditionally, in order to achieve this goal, such a user would apply Information Retrieval techniques such as those described in [SMcG83]. However, these techniques most often rely on **exact** keyword matching, and do not take into account the fact that the keywords may have some *semantic proximity* between each other. Let us stress that we are running the clustering on the sets of terms that describe the document, and that this list can be quite short. For instance a document might be characterized by the words “cat, food” and another with the word “feline, menu”. By using traditional methods these documents would be judged unrelated, however we will show that by using a distance between terms in an ontology, we are able to compute a meaningful similarity measure.

B. Related Work

THESUS system
We refer to [HVN+02, VN+02] for a longer version of this paper and a more detailed description of the THESUS system, including many experimental results.

Document Clustering
There has been large interest in this problem in the Information Retrieval community [Fis87,AGY99]. Our problem is more specific since we are clustering Web documents [ZE98], and take *links* into account [Kle99]. In order to cluster our documents we an algorithm based on incremental DB-Scan [EK+98].

Similarity Measures
In order to cluster the documents we need to use a similarity measure between sets of elements of a hierarchy (ontology). We are unaware of any specific work in the field, and have as such devised our own similarity measure. Our work is based on the Wu and Palmer [WP94] similarity between single elements of a hierarchy. In the general literature there has been interest for the topic of distances between sets of points [EM97,GHO+96]

Related Systems
- Kartoo [Kar] : a meta search engine that exploits links
- Vivissimo [Viv] : a clustering engine that uses the output of a search engine.

- Haliwala *et al.* [HGI00] also present a system that uses links to cluster web pages together, but do not use the semantic approach we have here.

C. Preliminaries

In this section, we explain how we create the ontology, and how we get the initial set of documents.

Ontology creation
In order to provide semantic clustering functions, we need to refer to an ontology of terms that are relevant to our domain of interest: the ontology used is based on DMOZ, manually trimmed and rearranged.

In what follows, we assume that the ontology is a **tree**, for the purpose of applying various similarity measures.

Document acquisition

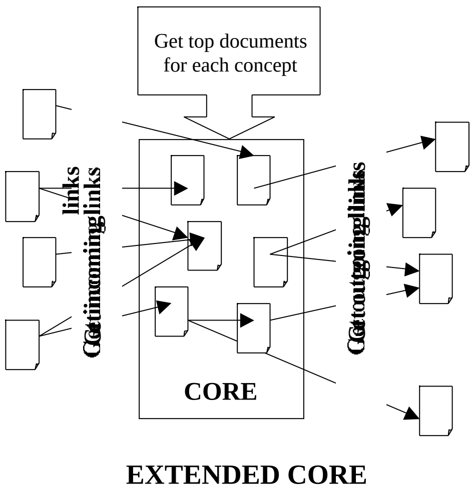

Figure 1. Document Acquisition Extended Core

We use a document acquisition module, whose goal is to create a collection of web documents that possibly relate to the predefined ontology. In order not to bias the pages collected, the acquisition process starts from an initial core of documents and expands it with the documents that they point to. The concepts (C_1,C_{11},C_{12},…Cm) of the ontology are used for the generation of the core in the following manner:

Given that the ontology has a tree structure, we generate a string (C_1+…+C_i) for each concept (Ci) that represents the path from the root to that concept. Then the path of each concept is used to query a web search engine by using the conjunction of all words that appear in the path. We retain the top *n* documents that contain all the concepts in the path. Therefore an ontology of *m* concepts will result in *m* queries and will generated a core that contains a maximum of *m*x*n* documents. However, the number of documents in the core is smaller since certain documents may appear more than once. Then the core is extended: first with the documents that are pointed to by the documents in the core (outgoing links) and secondly by the documents that point to the documents in the core (incoming links – they can be collected using a service provided by many web search engines, Google being an efficient one).

Other precalculated operations:
In order to speed up a number of operations, the Wu and Palmer[WP94] similarity between all the elements of the ontology will be pre-calculated. This is a table *m*x*m*, where *m* represents the number of terms in the ontology. We assume that the distance can be stored in a 4 byte float. This means the table takes under 1MB in main memory, which is quite reasonable. Since these similarities will be used extensively, especially during the clustering algorithm, it is important to keep them cached in this manner.

D. System Architecture

In this section, we will give the general architecture of THESUS. Refer to Figure 2 for its schema. We assume an initial set of documents D = {d_i}, that are somewhat related to a certain thematic domain of interest, and an ontology O on this thematic domain (e.g., Arts, Music, Technology, etc).

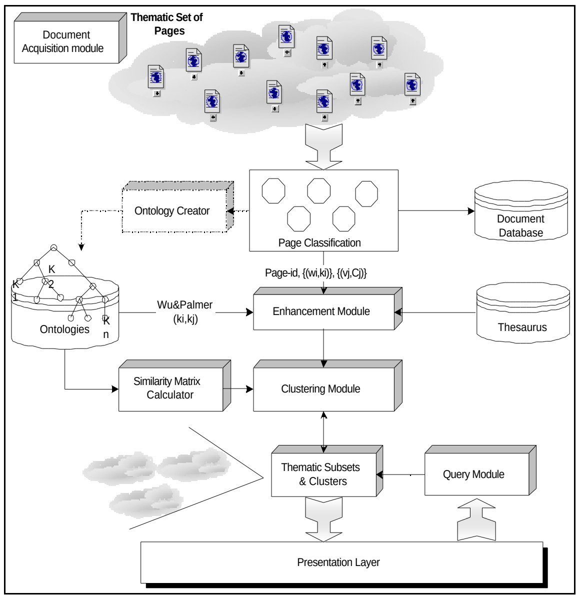

Figure 2. Proposed THESUS System Architecture

General explanation of modules

We can divide the operations THESUS will perform into different modules (see Figure 2).

1. Enhancement module: this module enhances each document in the initial thematic set, by extracting keywords to characterize the page. These keywords are then mapped to categories of the ontology of the domain.
2. Clustering module: The pages are clustered, and each cluster is labelled.
3. Query module: we can apply simple keyword based queries to our warehouse.

These three modules are connected with a database in which we store the document, the thesaurus, and the ontology.

D.1. Enhancement module:

The documents (otherwise pages) are processed in order to enhance their classification. The first step is to process the incoming links to D and extract keywords from the respective source pages. Each keyword k_i is mapped to a category c_j of the ontology O, using a thesaurus (WordNet [WorNet]). A similarity measure, the Wu & Palmer distance, will be applied to measure the degree of similarity between the keywords and categories. The outcome of this process is that every document in the original set will be enriched with:

- Keywords and weights (indicating the relevance between the keyword and the page)

- Categories of the ontology into which each document falls and weights. (defined by the keyword weights and the distances on the Ontology).

We use the following notation to define these **enhanced documents**:

Definition: an **enhanced document** is the triplet (Doc, K, C), where Doc is the document itself (or some way of accessing it, such as its URL), K (resp.C) is the set of couples $\{(w_i,k_i)\}$ (resp. $\{(v_j,c_j)\}$) of weighted keywords (resp. weighted categories) that define the document (w_i (resp. v_j) is a real, k_i (resp. c_j) is a string). Note that CURSIVE CAPITALS indicate weighted sets of words.

The option to integrate in this module a parser to extract words from the document itself and process them will be also taken under consideration in order to provide openness to other systems interconnection. Since the clustering module uses sets of weighted keywords, regardless of how these sets were constructed, this will be straightforward.
We will focus on how we extract the keywords from the links further down.

Reverse Link Keyword Extraction:
In THESUS system, keywords will be extracted from
- the link anchor (i.e. the string between the <a> and </a> tags) and
- from two text strings (each of 100 characters long), one preceding the starting link tag and another following the ending link tag.

The "window" will be trimmed whenever certain html tags appear, such as <a>, <li>, <tr>, <td> etc, because such tags usually indicate the logical end of the hyperlink neighboring area. As a result the mean number of keywords extracted for each hyperlink is less than five words.
Every keyword extracted will be assigned with a weight. We are currently exploring how to best extract this weight.

Finding related concepts:
Until this step, we have enhanced the document with some keywords that define what the document deals with. Yet, THESUS goes even further by defining a similarity measure between words, based on an ontology and a thesaurus. Given the thesaurus and the ontology from pre-processing components we can find for every word defining our document, which category of the ontology it is closest to. The similarity between a keyword k_i and the closest category c_i of the ontology is seen as the weight v_i of this category in the set of categories that defines the document.

The output of this module is an **enhanced document** as defined in previous paragraph. Note that the *Doc* entry of the triplet can be replaced with the URL of the document.

D.2. Clustering module:

In this phase, the documents are fed into the clustering module. The clustering algorithm will be based on a similarity measure between sets of weighted words. This function will be able to measure the similarity between the sets of keywords (resp. sets of categories), and also to measure the distance between a keyword and a category.
Here a task is the definition of a distance between **enhanced documents**. In fact, this boils down to finding the distance between set of weighted strings.

THESUS will combine the use of a similarity measure between elements of the set to calculate a more accurate similarity between the sets themselves. [EM97] proposed an interesting study of different measures between sets, and evaluates their complexity. For the algorithms proposed, the complexity ranges from polynomial (in the number of words in each set) to NP. Thus to be competitive, THESUS algorithm needs to have a polynomial behavior with regards to the number of elements in each set. THESUS will propose a novel similarity measure that is relevant to this problem. This similarity measure will be a generalization of Wu & Palmer [WP94] to sets of elements in a hierarchy. Using the notations given further up A and B are two sets of weighted keywords or concepts: $A=\{(w_i,k_i)\}$ and $B=\{(v_i,h_i)\}$

$$\boxed{\zeta(A,B)=\frac{1}{2}\left(\frac{1}{K}\sum_{i=1}^{|A|}\max_{j\in[1,|B|]}\left(\lambda_{i,j}\times S_{W\&P}\left(k_i,h_j\right)\right)+\frac{1}{H}\sum_{i=1}^{|B|}\max_{j\in[1,|A|]}\left(\mu_{i,j}\times S_{W\&P}\left(h_i,k_j\right)\right)\right)}$$

Where $\lambda_{i,j}=\frac{w_i+v_j}{2\times\max\left(w_i,v_j\right)}$ and $K=\sum_{i=1}^{|A|}\lambda_{i,x(i)}$

and

$$x(i)=x\,|\,\lambda_{i,x}\times S_{W\&P}\left(k_i,h_x\right)=\max_{j\in[1,|B|]}\left(\lambda_{i,x}\times S_{W\&P}\left(k_i,h_j\right)\right)$$

In an analogous way we define H and μ. We do not have the place to detail the similarity measure here, we refer to [HVN+02] for full details and examples.

Web Document Clustering Algorithm

Clustering aims at organizing patterns into groups, allowing us to discover similarities and differences, as well as to derive useful conclusions about them [HBV01]. The module will cluster web documents in order to discover meaningful groups. The problem is considerably different compared to the case of points in a metric space: in our case, the objects to be clustered are sets of (weighted) strings that correspond to categories of a domain ontology.

In this space there are no coordinates and ordering as in a Euclidean metric space. We only have a similarity measure (as defined in the previous section) between sets of (weighted) categories. We adopt this similarity measure. The clustering algorithm will be a modification of the density criteria used in popular density based algorithm DBSCAN.

The input of the clustering algorithm will be:
- the sets of weighted keywords characterizing each document,
- a threshold for the definition of the neighborhood, *MinSim*
- the minimum number of documents in the neighborhood of a document, *MinDocs*.

The output will be:
- the **partitioned set of documents** with respect to *MinSim* and *MinDocs*, and
- a set of documents, which are considered as noise.

Once the clusters have been found, a very important issue is their labelling (i.e., the assignment of a succinct yet descriptive set of categories to each cluster in order to facilitate user navigation and querying). A cluster Γ_i is defined as: $\Gamma_i = \{c_j\}$ where for every j, c_j is a category of ontology O. The output of the clustering module is the set of enriched documents, along with the cluster id to which each document belongs.

Grouping documents together is itself a semantic enhancement. We would also like to find appropriate labels for each cluster for the following reasons:
- Simply grouping documents together does not give a way of characterizing this set.
- We need some sort of way of calculating which cluster a given query is most closest to.
- Giving a more precise characterization to the cluster will enable easier browsing through the set of documents as a whole.

In this context two questions will be answered during the System Design Work package:

What sort of label?

THESUS will deal with ways to process sets of keywords, or weighted sets of keywords. It seems pretty straightforward to try to construct labels of this type for every cluster generated. This would in particular reduce the task of finding on which cluster we should apply a keyword query, to a simple similarity measure between that query (i.e. the set of keywords) and all the labels of the clusters.

How to construct such a label?

We want these labels to have some sort of significance. THESUS will consider two approaches on this task:

a. without weights:
- Construct U, the union of all concepts that appear in at least one document of the cluster.
- For every concept C_i in U, calculate the number of documents in the cluster that it appears in.
- Keep as a label for this cluster all the concepts C_i that appear in at least T% of the documents of the cluster where T is a threshold value.

b. with weights:

It is pretty straightforward to assign as weights the percentage of documents of the cluster that are characterized by a given keyword.

<u>D.3. Query module:</u>

The clusters produced by the previous module can be exploited to answer user queries, in a more meaningful way. In THESUS, we consider queries to be simple keyword based queries. Let us briefly explain the query mechanism, that we will facilitate THESUS system.
Let $q=\{k_i\}$ represent a query, where k_i are keywords defining the user's interest. THESUS will first of all identify the cluster(s) relevant to *q*. The irrelevant clusters are pruned and will not be considered in the search. The query-processing module then proceeds to determine within the selected cluster(s) the documents most relevant to the query, and can present the results ordered, and classified.

Query Processing Algorithm
In order to provide answers to a keyword based query, THESUS will have to perform the following actions:
<u>- Map the keywords of the query to the ontology</u>
Using similarity measres, we can map each keyword to a word in the ontology, in the same way as we do for the clustering module. The Wu & Palmer similarity between a keyword and the category of the ontology is seen as the weight of that category. We will only keep categories that have a weight above a certain threshold.

<u>- Find which cluster(s) is(are) closest to the categories that now define our query.</u>
We will have to calculate the similarity of the query, that is now a set of weighted categories of the ontology with all the labels of the clusters. Since the Wu & Palmer distance between categories of the ontology is cached, this operation will be very fast This prunes out a lot of clusters to make the next steps faster.

<u>- Calculate the similarity between the keyword query, and every document in the selected clusters.</u>
To do this, we have to consider their keyword description, since it is more precise. Once again, we will use here the similarity measure between sets of (weighted) words, but this time the Wu & Palmer distance is not cached (THESUS will not precalculate all the similarities of words in thesaurus).

<u>- Return the </u>documents that are the closest in terms of similarity.
For every document returned by the query, we have a similarity measure, so we can rank the results. We also present the results within the clusters they were found in, to help users browse results faster.

E. Conclusion

In this article we present the architecture of a system, THESUS, designed to cluster Web pages together, using incoming links to better describe pages, and an ontology of the domain. We give some experimental results in [HNV+02].
Future Work:
- We plan on improving the way in which we construct the hierarchy of clusters. We are currently experimenting using the CobWeb algorithm to this end.
- The query system can be made more efficient. This involves improving the mapping techniques that we use in order to determine which word of the ontology a generic word should be mapped to. This is a topic that we are currently working in, and that is very closely linked to this article, since it is used to define the small sets of terms that define a web document.
- We will investigate storage and query optimization, using M-Trees, that are adapted to our problem.

Acknowledgements
We would like to thank the following people for their helpful discussions on various topics related to this paper: G. Cobena and S. Abiteboul (general work on the SPIN projet [ANC+02], a similar effort to construct a personal thematic warehouse). Ch. Froidevaux, C. Nicaud, K. Nørvåg, B. Safar and L. Segoufin(similarity measures and distances). J.P. Sirot (ontologies). P. Rigaux and P. Veltri (R*-Trees).

F. Bibliography

[ANC+02] Serge Abiteboul, Benjamin Nguyen, Gregory Cobena and Antonella Poggi, *"Construction and Maintenance of a Set of Pages of Interest (SPIN)"* to appear in Bases de Donnees Avancees (2002)
[AGY99] Charu C. Aggarwal, Stephen Gates and Philip Yu, *"On the merits of building categorization systems by supervised clustering"*, Proceedings of the 5th ACM-SIGKDD p.352-356 (1999)
[EK+98] Martin Ester, Hans-Peter Kriegel, Jorg Sander, Michael Wimmer and Xiaowei Xu, *"Incremental Clustering for Mining in a Data Warehousing Environment"*, Proceedings of the 24th VLDB Conference (1998)
[EM97] Thomas Eiter, Heikki Mannila, *"Distance measures for point sets and their computation"*, in Acta Informatica Journal, 34, 109–133 (1997)
[Fis87] Douglas Fisher, *"Knowledge Acquisition via Incremental Conceptual Clustering"*, in Machine Learning 2: p139-172 (1987)

[GHO+96] J. Green, N. Horne, E. Orlowska and P. Siemens, *“A Rough Set Model of Information Retrieval”*, Theoretica Infomaticae 28, pages 273-296 (1996)

[HBV01] M. Halkidi, Y. Batistakis, M. Vazirgiannis, *“On Clustering Validation Techniques”*, Intelligent Information Systems Journal, Kluwer Pulishers, 2001

[HGI00] Taher H. Haveliwala, Aristides Gionis and Piotr Indyk, *“Scalable techniques for clustering the Web”*, in proceedings of Webdb 2000 worshop

[Kar] http://www.kartoo.fr

[Kle99] J.Kleinberg, *“Authoritative sources in a hyperlinked environment”*, Journal of the ACM 46(1999).

[VN+02] I. Varlamis, B. Nguyen, M. Vazirgianis and S. Abiteboul, *“Effective Thematic Selection in the WWW based on Link Semantics”*, Technical Report (2002)

[HNV+02] M. Halkidi, B. Nguyen, I. Varlamis and M.Vazirgiannis, *“THESUS: Organising Web Document Collections Based on Semantics and Clustering”*, Technical Report (2002)

[SMcG83] Gerard Salton, Michael McGill, *“Introduction to Modern Information Retrieval”*, McGraw-Hill, New-York (1983)

[Viv] http://www.vivisimo.com/

[WorNet]http://www.cogsci.princeton.edu/~wn/

[WP94] Z. Wu and M. Palmer *“Verb Semantics and Lexical Selection”*, Proceedings of the 32nd Annual Meetings of the Associations for Computational Linguistics, pages 133-138.

[ZE98] Zamir, Etzioni, *“Web document clustering: a feasibility demonstration”*, in proceedings of ACM-SIGIR ‘98