THESUS: Organizing Web Document Collections Based on Semantics and Clustering

Maria Halkidi[1], Benjamin Nguyen[2], Iraklis Varlamis[1], Michalis Vazirgiannis[1,2]

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

[2]INRIA Rocquencourt
Domain de Voluceau
78153 Le Chesnay
FRANCE
Firstname.Lastname@inria.fr

Abstract

The requirements for effective search and management of the WWW content are stronger than ever. Document retrieval in the WWW is problematic with regards to three aspects: i. Not taking adequately into account link semantics for characterizing documents ii. Basing document similarity on binary keyword matching, thus the matching of different yet semantically close keywords would return false. iii. Presenting search results in terms of lengthy URL lists. The previous observations lead us to the adoption of an ontology and a thesaurus to exploit links, in order to provide a better characterization of web documents. The enhancement of the characterization of the documents makes operations such as clustering become very interesting. We have devised a system called THESUS to achieve these goals. The system deals with an initial set of web pages, extracts keywords from all pages' incoming links and converts them to semantics by mapping them to a domain's ontology. Subsequently, a clustering algorithm is applied to discover document clusters. The clustering results are used for: i. Web pages classification ii. speeding up (in terms of pruning) keyword based queries. The system is fully functional and testing proved the quality of its results. The innovations lie in the integrated use of: i. incoming link semantics as the fundamental document characterization mechanism, ii. the conversion from keywords to semantics and iii. the clustering of web content using the semantic similarity between documents.

1 Introduction

"Birds of a feather flock together"
--Old English adage

In many different fields of science, ranging from sociology to physics, there is the common paradigm that an entity is more than the sum of its parts. However, when applied to searches on the web, the traditional process is to consider that a page is defined exclusively by its content. Traditionally, World Wide Web surfers submit queries to search engines by using one or more keywords to define their interests. In such search engines (Google [Goo], Lycos [Lyc], Alta-Vista[Alt]), *a web page's semantics are implicitly* defined by the keywords that are assigned to the page, usually because they appear in the body of the page. It is only recently that Google has taken into account words contained in hyperlinks that point to a page to enhance search results. Moreover, the similarity between web pages is based on *exact* matching of the keywords that define the page, and the query. [GGK01] shows that by considering *"…web pages as the sets of words they contain, a clustering operation based on set similarity could identify clusters of web pages, which are similar but not copies of each other. Such an operation would be very useful for a more efficient (and accurate) implementation of "what's related" feature of popular web browsers".*

Query results are also hard to tally, since search engines usually return huge lists of URLs, most of which can often be judged irrelevant to the query. We feel there exists a requirement for organizing the search results into meaningful groups according to their semantic similarity. This organization will benefit the user in terms of navigation through thematic groups, rather than linearly browsing a list of pages.

Motivation for link use:

The World Wide Web is a graph. More precisely still, it is a directed labeled graph, where the pages represent the nodes, and the links between pages represent the edges of the graph. Though link analysis and link structures are exploited in order to rank the importance of pages [Google, Kle99] the links' semantics are generally ignored. We believe that we will be able to extract crucial semantic material about a web page from the pages that point to that page. In this article, we will show how to add semantic labels to the edges of the graph of the Web, and use them for both document clustering, and ranking results of queries.

Motivation for ontology use:

Web documents are mainly characterized by extracted keywords and by a rank that takes into account link structures [BP98]. Finding the similarity between documents is based on exact matching between these terms; which is hardly similarity… it is binary matching! For

instance, a document d_1 characterized by the keyword list: d_1={snake, desert} would be judged irrelevant to a document d_2 characterized by the list: d_2={adder, Sahara} yet it is arguable that the two lists of keywords (and thus the documents) are in fact related, since an "adder" is a "snake" and the "Sahara" is a desert, therefore d_2 deals with the same concepts as d_1, they are just more specialized.

Motivation for web documents clustering:

Popular search engines usually return a (long) ranked list of URLs. To find a page exactly fitting her needs, a user needs to browse the list of URLs until she finds a URL of interest. Lately, a large work effort has been concentrated on improving the ranking of search results, with sometimes impressive progress. Yet, one-dimensional ranked lists are inherently limited. The grouping of URLs into meaningful sets, along with an appropriate labeling of these sets, and a flexible means of navigation within and across these sets would greatly assist the user in his task of information retrieval. We claim that document clustering -the automatic organization of documents into meaningful groups, see [Viv]- is an essential tool that needs to be fully exploited in web content organization and searching.

Observations:

Document retrieval in the WWW is problematic with regards to three aspects: i. Not taking adequately into account link semantics for characterizing documents ii. Basing document similarity on binary keyword matching, i.e., the matching of different yet semantically close keywords would return false iii. Presenting search results in terms of lengthy URL lists.

Main idea:

The previous observations and claims lead us to the adoption of ontologies[Gua98] to exploit links, in order to provide a better characterization of web documents. The enhancement of the characterization of the documents make operations such as clustering highly interesting.

Overall contribution:

A system (THESUS) that enhances the semantics of web documents and enables more efficient searches by *exploiting* them in an integrated manner.

Specific contributions:

i. Use of link semantics as the main feature for page classification.
ii. Use of an ontology and thesaurus for semantic matching between documents and keyword based queries –thus avoiding the curse of exact keyword match.
iii. A novel similarity measure for web document clustering, also usable in the general context of similarity between of sets of elements of a metric space.
iv. A clustering algorithm that discovers meaningful groups of pages in a set of pages from the web. These are thematic subsets, that we call THESUs.

Restriction: We do not consider in this article the temporal evolution of these sets of documents, but this remains an axis of our current research.

THESUS is NOT a search engine but rather a system that enhances the semantic organization of a set of pages and guides the user within a set of pages. It is envisioned as a personal service that will assist the user in the creation and querying of rather compact high quality thematic collections of pages. Of course the concepts developed here are expandable to the full WWW; we bore scalability in mind during the creation and the implementation of THESUS.

The system is implemented in Java, and in the process of fine tuning. In this paper, we present the architecture of the system, the basics of every module of the system, and convincing results produced by the system.

Note: THESUS is the name of the system. A THESU is a thematic subset, and the plural is THESUs. We refer indifferently to a document or a page to define the basic entity of our system.

The organization of the paper is as follows: Section 2 presents the work related to our project. Section 3 details the THESUS system. Section 4 presents the results of the system.

2 Related Work

In this section we review related concepts and work in terms of i. Techniques and methodologies used in our system and ii. Related approaches and systems.

2.1 Useful concepts and techniques

Ontologies and Thesauri

In our system, we rely on the use of an ontology and a thesaurus (WordNet), in order to compute similarities between words. For more details on ontologies you can refer to [Gua98] and for information on WordNet [WorNet, ABF+98]

Link analysis and semantics

Some works relate to the importance of links as entities that promote semantics in a hypermedia network. Kleinberg [Kle99] states "*The link structure of a hypermedia environment can be a rich source of information about the content of the environment (...) But for the problem of searching in hyperlinked environments such as the World Wide Web, it is clear from the prevalent techniques that the information inherent in the links has yet to be fully exploited*".

Link information is already used by web search engines to filter and rank query results better, e.g Google's PageRank algorithm prioritises pages with many incoming or outgoing links. Google's PageRank algorithm associates the text of a link with the page that the link points to, thus providing more accurate descriptions of web pages than the pages themselves, which, as a matter of fact, need not to be fetched.

ARC system [CD+98] uses the text around *href* links to describe the contents of the pointed page and to enhance the hub and authority weight on a certain topic. In particular, if text descriptive of a topic occurs in the text

around an href into a page *p* from a good hub, it reinforces the belief that *p* is an authority on the topic. The main assumption is that links to a certain page act as recommendations of this page by authors of other pages. Hopefully, these authors act independently of the page's author and their opinions can be added to the ranking of the page. According to [Hen01], this kind of ranking is called connectivity-based ranking.

Distance/similarity measures between sets

In [EM97] the problem of measuring the similarity or distance between two finite sets of points in a metric space is considered. Some of the distance functions are reviewed, amongst them the minimum distance link measure, the surjection measure, and the fair surjection measure. All of them take polynomial time algorithms for their computation. The indexing issue for distance / similarity between sets of values is treated in recent work [GGK01] where similarity between two sets of values is considered as the ratio of common elements to the union of the values of the two sets (Jaccard coefficient).

In all the aforementioned efforts, the sets of values are considered as flat (i.e. all values are independent from each other) and only exact matching between values is taken into account. By mapping keywords to categories of the ontology, and calculating similarity between these categories, documents defined by different sets of words may end up grouped together. [BSF02] investigate this with their mediator approach.

Basic Clustering Concepts:

The clustering problem is about partitioning a given data set into groups (clusters) such that the data points in a cluster are more similar to each other than points in other clusters [GRS98]. In the last years the availability of huge transactional and experimental data sets and the arising requirements for data mining created needs for clustering algorithms that scale and can be applied in diverse domains. A multitude of clustering methods are available, which can be broadly classified into the following types [HBV01]: i. *Partitional clustering* attempts to directly decompose the data set into a set of disjoint clusters. ii. *Hierarchical clustering* proceeds successively by either merging smaller clusters into larger ones, or by splitting larger clusters. iii. *Density Based clustering.* The key idea of this type of clustering is to group neighboring objects of a data set into clusters based on density cluster conditions.

Also, depending on the type of variables allowed in the data set, the methods can be categorized into [HBV01]:

- *Statistical*, which are based on statistical analysis concepts. They use similarity measures to partition objects and they are limited to numeric data.
- *Conceptual*, which are used to cluster categorical data. They cluster objects according to the concepts they carry.

Information Retrieval Techniques

The problem of searching for documents to answer a query and managing text documents is typical of *Information Retrieval* (IR). A number of IR techniques have been proposed such as Vector space retrieval (See [Sal87, SMc83]), Probabilistic retrieval and Relevance Feedback (See [RS-J76, Rij79, CH79]).

2.2 Related systems

There are several related efforts in the area of web document clustering, both research and industrial

Industrial

Vivísimo [Viv] proposes a clustering approach for web document organization. It makes use of the contents (titles and brief descriptions) that are returned by the underlying search engines. Northern Light [Nor] classifies each document within an entire source collection into pre-defined subjects and then, at query time, selects those subjects that best match the search results. Vivísimo does not use pre-defined subjects; its annotations are created spontaneously.

An interesting server that uses hyperlink's structure to group interconnected results is Kartoo [Kar]

Meta search engines (such as Copernic [Cop]) collect results from various search engines and group them according to their relevance. But they do not use ontologies to relate documents whose keywords do not match lexicographically.

Research

Here, for space reasons, we will constrain our review to **web document clustering** approaches. There is a consensus that clustering techniques should be applied to the results of a query rather than on the whole web space, in order to discover groups of relevant documents. In [ZE98] a suffix tree-clustering (STC) approach is proposed where the algorithm based on phrases shared between documents is used to create the clusters. The documents to be clustered are first cleaned (using a stemming algorithm). Then the base clusters are identified using the suffix tree structure. This process can be viewed as the creation of an inverted index of phrases for the document collection. Finally the significantly overlapping base clusters are merged using a similarity measure that is based on the number of documents common to two clusters. The results are encouraging from the efficiency point of view and the algorithm is linear in the number of documents.

Gionis et al. [HGI98] propose a clustering approach for Web documents to find groups of similar pages. The similarity is based on words found in the documents along with their occurrence frequencies. The similarity measure adopted represents the ratio of the common terms divided by the union of the terms of two documents. Term matching is exact.

Criticism

These systems / algorithms suffer from two problems: i. they do not seem to exploit specifically link semantics and ii. they only deal with exact matching of keywords for documents' similarity computation, which eliminates the similarity that can be acquired from an ontology.

3 THESUS System

In this section, we will describe the general goals of our system, then detail its architecture, and the various modules that compose it.

3.1 Objective

Our primary goal, when designing THESUS was to create a system that could enhance the semantic structure and organization of a set of documents (mostly in the WWW context, such as a web warehouse) in order to increase the quality of search (both navigation and result) amongst these documents. We achieve these objectives by the integrated use of inverse links, an ontology, a thesaurus, and clustering.

3.2 Architecture

In this section, we give the global architecture of THESUS. Refer to Figure 1 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 (i.e., Arts, Music, Technology, etc). In order to test out THESUS, we used results from search engines to construct D. We will give more details about this in Section 3.3.

General explanation of modules:

We can divide the operations THESUS performs into different modules (see Figure 1).

a. 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.
b. Clustering module: The pages are clustered, and each cluster is labelled.
c. Query module: we can apply simple keyword based queries to our warehouse.

These three modules are connected with an SQL database in which we store the documents, the WordNet 1.6 database, and the ontology. The system is implemented in Java, and has been tested on different sets of the order of 10^4 documents, a reasonable size for a personal thematic warehouse. For more results, see Section 4.

Enhancement module:

The documents (otherwise pages) are processed as follows 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, in our case Wordnet [WorNet, ABF+98]. We use a similarity measure, the Wu & Palmer measure [WP94] to measure the degree of similarity between the keywords and categories. See Section 3.5.a. for a precise discussion on this measure and why we chose it. The outcome of this process is that every document in the original set is now enriched with:

i. Keywords and weights (indicating the relevance between the keyword and the page)

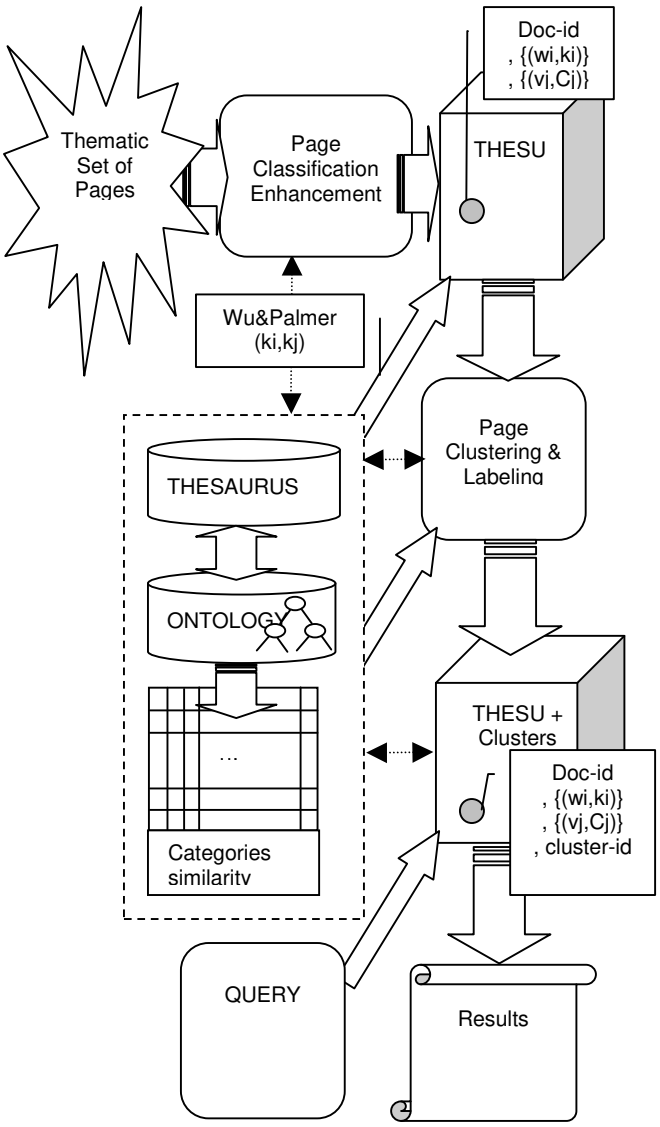

Figure 1. THESUS functional architecture.

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

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

Definition: an **enhanced document** is the triplet (Doc, $\mathcal{K}$, $\mathcal{C}$), where Doc is the document identifier (such as its URL), $\mathcal{K}$(resp.$\mathcal{C}$) is the set of couples $\{(w_i,k_i)\}$ (resp. $\{(v_i,c_i)\}$) of weighted keywords (resp. weighted categories) that define the document (w_i (resp. v_i) is a real, k_i (resp. c_i) is a string). Note that w_i and v_j are not necessarily the same if i=j.

The weights are a measure of relevance of the keyword or concept to the document. A weight is a real from the interval [0;1]. A value of 1 indicates total relevance, 0 indicates no relevance. A value of 0.2 would indicate slight relevance.

Remark: these keywords are a *positive* definition of the document. We do not *negatively* characterize the document

(for instance with words that we would say are *definitively not* relevant to the document).

Clustering module:

In this phase, the documents are fed into the clustering module. The clustering algorithm is based on a similarity measure between sets of weighted words. It is defined in Section 3.5.a. This function can measure the similarity between the sets of keywords (resp. sets of categories), and can also measure the similarity between a keyword and a category.

Once the clusters have been found, a very important issue is their labelling (i.e., the assignment of a succinct 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.

Query module:

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 detail further in Section 3.6.a. 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.

Important observation: The initial set of documents is not expanded throughout the THESUS processing: no new documents are added. In each phase additional semantic tags (classes, keywords, cluster ids) are attached to the documents, increasing the semantic organization and content of the document set. The enrichment of the set of documents by discovering on the web other documents that may be of interest is beyond the scope of this paper.

In the following sections, we will describe with more detail the algorithms used in the various modules of THESUS.

3.3 Preliminaries

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

a. 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. We do not provide an automatic ontology creation. Instead, we assume that an ontology has already been constructed. In our prototype, we used an ontology based on DMOZ [Dmo], that we manually trimmed and rearranged. In what follows, we assume that the ontology is a **tree**, for the purpose of applying our various similarity measures. In our example, the ontology thematic was 'technology' and contained 467 different categories (words).

b. Document acquisition :

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 categories (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 category (C_i) that represents the path from the root to this category. Then the path of each category is used to query a web search engine by using the conjunction of all categories that appear in the path. We keep the *n* top-ranked documents that contain all the categories in the path. Therefore an ontology of *m* categories will result in *m* queries and will produce a core that contains a maximum of *m*x*n* documents. However, the number of documents in the core is smaller, since many documents appear multiple times.

Then the core is extended (See Figure 2): first with the documents that are pointed to by the documents in the core (outgoing links) and then by the documents that point to the documents in the core (incoming links – they are collected using a service provided by many web search engines, in our case, we used Google).

For the 'technology' ontology, the core contained approximately 39,000 distinct documents whereas the extended core contained 1,600,000 distinct documents. It is interesting to notice that approximately a million of these documents were pointed to by at least one of the documents in the collection. For these documents we can have incoming hyperlink information, extracted by the documents in the collection.

c. Other precalculated operations: Wu and Palmer similarity caching.

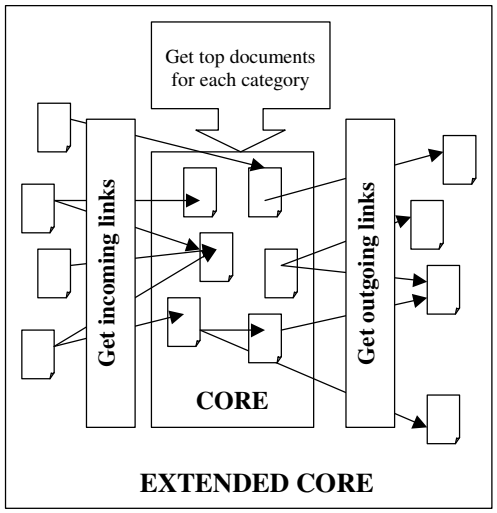

Figure 2. Core collection extension mechanism

In order to speed up a number of operations, we precalculate the Wu and Palmer[WP94] similarity between all the elements of our ontology. This is a *m*x*m* table, where *m* represents the number of terms in our ontology (467 in our case). We store the similarity in a 4 byte float. This means the table takes under 1MB in main memory, which is quite reasonable. Since we use these similarities a lot, especially during the clustering algorithm, it is important to keep them cached in this manner.

3.4 Enhancement module

The goal of this module is to enhance the semantics of a document, to improve clustering and querying. This can be viewed as adding some meta information to each document in our database. Our idea is to find a set of words that will correctly define what a document deals with. We have chosen in this paper to focus on incoming links, using techniques we formerly proposed in [VN+02]. In this previous paper, we have shown that incoming links provide a comparable -if not better- description to the processing of the text in a page. Nonetheless, we could integrate in this module a parser to extract words from the document itself and process them, since our clustering module uses sets of weighted keywords, regardless of how these sets were constructed.

We will focus on how we extract the keywords from the links further down.

The page information enhancement module:

- Extracts keywords from the reverse links of the page (i.e., pages that point to this page)
- Create a set of weighted keywords that define the page
- Using heuristics, finds a set of (weighted) keywords that characterize this page
- Maps the set of keywords that characterize the page, into a set of (weighted) categories (i.e. terms that appear in the ontology)

The result of this process is that the document is not only characterized by a set of weighted keywords but also by a set of weighted categories to which these keywords are related.

a. Reverse Link Keyword Extraction

The issue of extracting keywords from the links is important in the context of this paper. In [PW00] the idea of "robust" hyperlinks is introduced. Robust hyperlinks are defined as the ones that contain descriptive information on the target document. This information can be limited to five words according to the authors. In [CD+98] some experiments on the hyperlink neighboring text, result in defining the "anchor window" to be 50 characters. In THESU system, keywords are extracted from i. the link anchor (i.e. the string between the <a> and tags) and ii. from two text strings (each of 100 characters long), one preceding the starting link tag and another following the ending link tag. The "window" is trimmed whenever certain html tags appear, such as <a>, <li>, <tr>, <td> etc, because such tags usually signify 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. In the HTML fragment that follows, keywords are extracted from the hyperlink to the CERN homepage:

```
<A   HREF="http://cts-hp.cts.iisc.ernet.in/">
Centre for Theoretical Studies</A><D>Indian
Institute of Science, Bangalore,
India<DT></LI>                          <LI><A
HREF="http://www.cern.ch/">CERN</A><DD>
European Organization for Nuclear
Research/European Laboratory for Particle
Physics, Geneva, Switzerland (See also <A
```

From the hyperlink to the URL: http://www.cern.ch the phrases extracted are:

- The hyperlink area (dark color white text) that gives phrase "CERN",
- The 100 charaters preceeding (grey area) the hyperlink. This area is trimmed where <LI> appears so no characters remain.
- The 100 charaters following (grey area) the hyperlink. The last partially selected keyword ("Swi") is ignored, <DD> is removed, stopwords "for" and "de" are removed also. The remaining phrase is: "European Organization Nuclear Research/European Laboratory Particle Physics, Geneva".

So the set of keywords that describe this specific target URL is {cern, european, organization, nuclear, research, laboratory, particle, physics, geneva}, where the word "european" is counted once. In this way, we are able to acquire a set of related keywords for every document that points to a given document d_i. We do this for every document in our database.

b. Creation of the weighted set of keywords

In order to create the set of weighted keywords that define a document d_i, we merge all the sets of keywords that appear in the pages that point to d_i, while adding weights in the process. For every keyword extracted, the weight characterizes the credibility we have that d_i actually deals with that keyword. It is our intuition that the more incoming links that define the document with a given keyword, the higher the probability that the document indeed deals with this keyword. We use the following method to quantify these weights: Let *inc* represent the number of incoming links to d_i. Let $S_i=\{S_{i,j} | j \text{ in } [1;inc]\}$ represent the set of sets of incoming keywords that define document d_i. For every word in $S_{i,j}$ we add the word to the set of weighted keywords K_i that define d_i with weight w_i where w_i represents the number of different sets in which this keyword appears, divided by γ (a constant) or $w_i=1$ if this value is greater than 1. Intuitively, γ represents the minimum number of sets $S_{i,j}$ that must contain this keyword in order to be sure that it correctly characterizes the document. γ is a parameter, we propose a value of 5, but have not extensively tested the impact of this value.

According to [CD+98] all the words retrieved are highly relevant to the document. If we wanted to expand the window to include more characters, we would decrease the weight of the words retrieved within the larger window, to

denote lower credibility in their semantics. Thorough testing of this part of the system remains further research.

c. Finding related concepts

As we see now, we have enhanced the document with some keywords that define what the document deals with. If we simply query these links, even without using the text contained in the document itself, we get comparable results to commonly used search engines, see [VN+02].

Yet, we want to go even further. Rather than asking for an exact match in the keywords, we define a similarity measure between words, based on an ontology and a thesaurus. We assume both the thesaurus and the ontology are given. In our experiments, we used WordNet [WorNet, ABF+98] as the thesaurus, and we constructed an ontology based on DMOZ [Dmo]. Using WordNet, and assuming a similarity between words (Wu & Palmer[WP94]), we can find for every word defining our document, which category of the ontology it is closest to. (In this paragraph we use the notations defined in Section 3.2) 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. Note that if this weight is below a threshold (0.8 in our tests), we do not add the category to the list, since there is a possibility that the keyword was not very related to the category, and we would be introducing noise into the characterization of the documents. We then multiply this value by the corresponding weight of the keyword that mapped to this category, w_i. We can sum this up by the following formulas:

$$c_i = c_j \mid S_{W\&P}(c_j, k_i) = \max_k (S_{W\&P}(c_k, k_i))$$

and

$$v_i = S_{W\&P}(k_i, c_i).w_i$$

Note that in fact, we do not want to have multiple elements in $\mathcal{C}$ that contain the same category. For every couple $c_j=(v_j,c_j)$ in $\mathcal{C}$, we check to see if another couple $c_k=(v_k,c_k)$ corresponds to the same category (i.e., $c_j=c_k$). If that is the case, we eliminate from $\mathcal{C}$ the couple with the lowest weight.

The output of this module is an **enhanced document** as defined in Section 3.2..

d. Drawback

We used a WordNet function that tries to find for a given string which word in the database it is lexicographically closest to. In some cases, this results in a wrong mapping. We plan on investigating this problem.

3.5 Clustering module

In this section, we describe the algorithms used in the clustering module. First of all, to be able to run the clustering algorithm, we need a similarity measure over documents. A lot of work has already been conducted in this field, see [HBV01]. Then we will detail the clustering algorithm, which is based on the DBSCAN [EK+96] algorithm.

a. Similarity definition

First of all, we will explain how we characterize a document, then we will stress the properties that we require our similarity measure to have.

General ideas

We are faced with the task of defining a similarity measure between **enhanced documents**. In fact, this boils down to finding the similarity between sets of weighted strings. In the following paragraphs, we will propose the properties such a measure must have, and propose one that has them.

We have the same intuitions about the properties a similarity measure should possess as proposed in [Lin98]. We quote:

Intuition 1:The similarity between A and B is related to their commonality. The more commonality they share, the more similar they are.

Intuition 2: The similarity between A and B is related to the differences between them. The more differences they have, the less similar they are.

Intuition 3: The maximum similarity between A and B is reached when A and B are identical, no matter how much commonality they share.

This similarity measure will be used both when clustering the documents, and when answering queries. We expect it will be called upon very often, and thus it is essential that it runs as fast as possible, even on large numbers of documents. Therefore, we need to bear in mind the complexity of the calculation of this similarity. For scalability reasons, it must be independent of the number of documents in the database.

Intuitions on the calculation of similarity between weighted sets

A number of similarities/distances between sets of elements already exist in the literature[EM97,GHO+96]. The most widely used one, the Jaccard coefficient is simple: Let A and B represent two sets of elements. The similarity between A and B is defined by:

Jaccard similarity: $$S_1(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

This similarity respects the three intuitions given in the previous paragraph but it only takes into account exact matches. Yet in our situation we can go further: by introducing a similarity measure between words, we can define a measure that will take into account the proximity of words, when comparing sets of words, rather than a simplistic binary comparison. [RSM] and [Res95] propose different measure inside a taxonomy such as WordNet, and [Lin98] proposes a comparison between these measures and others, such as Wu and Palmer[WP94], Miller and Charles, and a novel similarity measure. Wu and Palmer measure is the fastest to compute, and is arguably as good as the others (see [Lin98]), thus we use it in our system. We detail its definition and properties in the following paragraph.

Wu and Palmer similarity measure

Given a tree, and two nodes *a,b* of this tree, we first of all find their deepest (in terms of depth in the tree) common ancestor *c*. The similarity measure is computed as follows:

$$S_{W\&P}(a,b) = \frac{2.Depth(c)}{Depth(a)+Depth(b)}$$

It is straightforward to check that $S_{W\&P}$ verifies the three assumptions of 3.5.a. Note that we consider the first node to be of depth 1 and not 0. As a side note, we can define a distance with this similarity:

The similarity between words is a real that ranges from 0 to 1. We can define the following distance measure on this similarity in the following way:

$$D = 1 - Sim$$

It is trivial to see that this distance verifies:

- D(x,y)=0 iff x=y
- D(x,y)=D(y,x)

It is straightforward but lengthy to show that this distance is a metric, i.e., that it verifies the triangular inequation.:

- D(x,y)≤D(x,z)+D(z,y)

A similarity measure only needs to verify the first two conditions (except that the first is replaced by S(x,y)=1 iff x=y). In the rest of this article we talk about Wu & Palmer as a similarity measure, and not a metric, the definition of a metric on sets of words remains future work.

The idea: Combining Wu & Palmer, and sets:

One contribution in this article is to combine the use of a similarity measure between elements of the set to calculate a more accurate similarity between the sets themselves. Let us stress that we can use any given similarity measure on the elements of the sets themselves. Further work would for instance involve devising a measure to take into account terms semantically close, like frequent item sets of IR. We could then use this (enhanced) similarity between words to be even more precise.

We are no longer limited by a binary relation (equality) on words. [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, our algorithm needs to have a polynomial behavior with regards to the number of elements in each set.

Let us not forget that we are calculating similarity between sets of *weighted* words, and not simply words, but for the sake of simplicity let us first consider the similarity between sets of non-weighted keywords. There has been very little research on creating a similarity measure on sets of elements of a space with a similarity measure defined see [EM97, Nii87]. In the next paragraph, we propose a novel similarity measure that is relevant to our problem. This similarity measure is a generalization of Wu & Palmer.

A proposal for a similarity measure between sets of words

Notations:

1. Let **Ω** represent a set of words.
2. Let $S_{W\&P}(a,b)$ represent the Wu and Palmer similarity measure between words a and b of **Ω**.
3. Let ζ(A,B) represent the similarity measure between subsets A and B of **Ω**. We call this similarity measure the **extended Wu and Palmer similarity measure.**

Remark: We say two documents are identical if and only if they are represented by the same subsets of **Ω**. We define ζ(A,B) in the following way:

$$\zeta(A,B) = \frac{1}{2}\left(\frac{1}{|A|}\sum_{a\in A}\max_{b\in B}\left(S_{W\&P}(a,b)\right)+\frac{1}{|B|}\sum_{b\in B}\max_{a\in A}\left(S_{W\&P}(a,b)\right)\right)$$

Note: |A| represents the cardinality of set A.

Properties of ζ(A,B):

1. It is straightforward to check that ζ(A,B) is a similarity measure: i. ζ(A,B)=1 iff A=B ii. ζ(A,B)=ζ(B,A)
2. This similarity extends the Wu and Palmer similarity, because the similarity between two sets A and B where |A|=|B|=1 is exactly the Wu and Palmer similarity between the (single) element both sets contain (i.e., Let A={a} and B={b}, then ζ(A,B)=$S_{W\&P}$(a,b).)
3. This similarity measure is calculable in O(|A|x|B|), assuming that the time to calculate $S_{W\&P}$ is O(1). This makes it comparable in terms of complexity to the measures proposed in [EM97].

A	**B**	**ζ(A,B)**
Cat, CD	Cat, CD	1.0
Cat, CD	Feline, Record	0.83
Cat, Tiger, Lynx	Feline	0.80
Microprocessor, Technology	Electronics, Systems, Development	0.65
Cleopatra	Queen, Egypt	0.60
Spy, Microfilm	Feline, Record	0.47
Book, Dog	Car, Rental	0.35
Spy, Microfilm	America, Revolution	0.11

Table 1. Wu and Palmer similarity between sets

A proposal for similarity measure between weighted sets.

To characterize our documents better, we do not use simply sets of words, but sets of weighted words. For example, a document that talks about the Beatles records could be characterized by the following weighted set $\mathcal{A}$={(1, "Beatles"), (1, "Records")}, meaning both keywords are equally very important, or $\mathcal{B}$={(1, "Beatles"), (0.2, "Records")} meaning the document mainly deals with the Beatles, and only a little about records. We discuss in Section 3.4.b. how we determine these values for each document. We say that two documents are identical if they are represented by the same weighted sets.

Notations:

1. Let **Ω** represent a set of words

2. We use cursive capitals to represent sets of weighted words, such as: $\mathcal{A}=\{(w_i, k_i)\}$, with $k_i \in \Omega$ and $w_i \leq 1$.
3. We use the corresponding regular capitals to represent the sets of keywords without weights associated with a weighted set:

$$A=\{k_i \mid \exists\, i, (w_i, k_i) \in \mathcal{A}\}$$

4. We note $\mathcal{A}=\{(w_i, k_i)\}$ and $\mathcal{B}=\{(v_i, h_i)\}$

Remark: $|A|=|\mathcal{A}|$ but we don't necessarily have $|A|=|B|$, i.e., the documents defined by $\mathcal{A}$ and $\mathcal{B}$ don't have the same number of weighted keywords.

We use the following similarity measure, whose properties we will discuss in the following paragraph. We define

$$\zeta(\mathcal{A},\mathcal{B}) = \frac{1}{2}\left[\frac{1}{|\mathcal{A}|}\left(\sum_{i=1}^{|\mathcal{A}|}\lambda_i \max_{j\in[1,|\mathcal{B}|]}\left(S_{W\&P}(k_i,h_j)\right)\right) + \frac{1}{|\mathcal{B}|}\left(\sum_{i=1}^{|\mathcal{B}|}\mu_i \max_{j\in[1,|\mathcal{A}|]}\left(S_{W\&P}(h_i,k_j)\right)\right)\right]$$

Where $\lambda_i = \dfrac{w_i + v_{x(i)}}{2.\max(w_i, v_{x(i)})}$ and

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

In a similar fashion, we define:

$$\mu_i = \frac{v_i + w_{y(i)}}{2.\max(v_i, w_{y(i)})} \text{ and}$$

$$y(i) = y \mid S_{W\&P}(h_i,k_y) = \max_{j\in[1,|A|]}\left(S_{W\&P}(h_i,k_j)\right)$$

Properties of $\zeta(\mathcal{A},\mathcal{B})$:

1. It is straightforward to check that $\zeta(\mathcal{A},\mathcal{B})$ is a similarity measure: i. $\zeta(\mathcal{A},\mathcal{B})=1$ iff $\mathcal{A}=\mathcal{B}$ ii. $\zeta(\mathcal{A},\mathcal{B})=\zeta(\mathcal{B},\mathcal{A})$
2. This similarity extends the similarity measure on sets of keywords, if all the weights in $\mathcal{A}$ and $\mathcal{B}$ are the same (i.e., $\mathcal{A}=\{(\chi, k_i)\}$ and $\mathcal{B}=\{(\chi,h_i)\}$). Moreover, let $\mathcal{A}=\{(1,a)\}$ and $\mathcal{B}=\{(1,b)\}$, we have the extension: $\zeta(\mathcal{A},\mathcal{B})=S_{W\&P}(a,b)$.
3. This similarity measure is calculable in $O(|\mathcal{A}| \times |\mathcal{B}|)$
4. We can apply this similarity measure on any other similarity measure between keywords, not just on the Wu and Palmer measure.

$\mathcal{A}$	$\mathcal{B}$	$\zeta(\mathcal{A},\mathcal{B})$
(1,Cat),(0.5,CD)	(1,Cat),(0.5,CD)	1.0
(0.5Cat),(1,CD)	(1,Cat),(0.5CD)	0.75
(1,Feline),(0.5,Cat)	(1.0, Cat)	0.70

Table 2. Wu and Palmer similarity between weighted sets

In our results, we used the weighted similarity measures. Finding a better (more accurate) similarity measure between weighted sets remains future research.

b. Web document Clustering algorithm

In this section, we detail the module that clusters 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. We only have a similarity measure (as defined in the previous section) between sets of (weighted) categories. We adopt this similarity measure for use in the clustering algorithm. Our algorithm is a modification of the density criteria used in the popular density based algorithm DBSCAN[EK+96]. We will discuss the differences between our algorithm and the DBSCAN algorithm in the following paragraphs.

We slightly modify four definitions given in [EK+96] in order to explain the algorithm.

Definitions

Definition 1: Enhanced documents $d_i(URL,\mathcal{K}_i,C_i)$ and $d_j(URL,\mathcal{K}_j,C_j)$ are neighbors if they satisfy the following condition.

$$\zeta(C_i, C_j) > MinSim,$$

where *MinSim*, is a user-defined threshold and $\zeta(C_i, C_j)$ is the similarity measure on weighted sets of concepts defined in the previous section.

Definition 2: Then the neighborhood of a document d_i can be defined as follows:

$$\text{Neighborhood}\,(d_i(URL_i,\mathcal{K}_i,C_i)) = \{\, d_k(URL_k,\mathcal{K}_k,C_k) \mid \zeta(C_i, C_k) > MinSim\}$$

Definition 3: *Density-reachable*: A document d_j is density reachable from d_i with respect to *MinSim* and *MinDocs* if:

a) The document d_i belongs to the neighborhood of d_j
b) The number of documents in the neighborhood of d_j is at least *MinDocs*, the minimal number of documents we want to have in a cluster. This is a user-defined parameter.

Definition 4: *Density-connected*: The documents d_j and d_i are density connected with respect to *MinSim* and *MinDocs* if there is a document d_k such that both d_j and d_i are density reachable from d_k with respect to *MinSim* and *MinDocs*.

The Algorithm

Clusters are defined as sets of *density-connected* documents.

To find a density-connected set, the algorithm starts with an arbitrary document d_i and retrieves all documents that are density-reachable from d_i with respect to *MinSim* and *MinDocs*. We call d_i a *core object* if there exists a density-connected set with respect to *MinSim* and *MinDocs*. In that case, a cluster is defined. If this is not the case, (i.e., no objects are density-reachable from d_i), then d_i is seen as *noise* and does not belong to any cluster. This procedure is iteratively applied to each document d_i in the database that has not yet been classified.

The input of the clustering algorithm is: i. the sets of weighted keywords characterizing each document, ii. a threshold for the definition of the neighborhood, *MinSim*,

iii. the minimum number of documents in the neighborhood of a document, *MinDocs*. The output is i. the **partitioned set of documents** with respect to *MinSim* and *MinDocs,* and ii. a set of documents, which are considered as noise. The partitioned set of documents is ready to undergo the cluster labelling phase.

Differences with the original DBSCAN algorithm

i. Different Contexts…

In [EK+96], DBSCAN is used in the context of spatial databases. The algorithm assumes that the elements to be clustered are points in a metric space, of a given dimension, usually 2 or 3. The distance measure used is a simple Euclidian distance. In order to compute the neighbourhoods of a document, the points are inserted into a R*-Tree [BK+90]. Note that the computation of the R*-Tree is not taken into account in the complexity of the DBSCAN algorithm.

In our case, we are not in a metric space whose dimension we know. We cannot use a R*-Tree. Instead, we simply precalculate the similarity between all the n documents in the set. We save them in n different lists of size n. This costs us $O(n^2)$. Then we sort each list, using Quicksort. This well known algorithm has an average complexity of $O(n \log n)$ to sort one list of length n. To sort our n lists, we have a complexity $O(n^2 \log n)$. Once this preprocessing phase is complete, we can apply the algorithm. Thanks to our preprocessing, running the algorithm will in fact have a comparable complexity to the use of R*-Trees.

ii. ...Same Time Complexity

The time complexity of the clustering procedure is based on the average complexity of defining density connected sets of documents, i.e., identifying the neighbors of the documents in the database. In our system, since the similarity of a document d_i with all other documents is pre-calculated in an ordered list, the time complexity of defining the list of the documents closest to it (i.e., with $\zeta > MinSim$) using a dichotomic method is $O(\log n)$. The algorithm does this task once for every document in the database, thus the complexity is $O(n \log n)$. Compared to the use of R*-Trees, Ester *et al.* claim that on average the complexity will be also be $O(n \log n)$.

c. Labeling the clusters

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

What sort of label?

Throughout this article, we have been dealing 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. We will detail this in a further section.

How to construct such a label?

Of course, we want these labels to have some sort of significance. We propose the following method to construct of a label. We feel that the results given in Table 3 justify the choice of this method.

i. without weights:

- We construct U, the union of all concepts that appear in at least one document of the cluster.
- For every concept C_i in U, we calculate the number of documents in the cluster that it appears in.
- We keep as a label for this cluster all the concepts C_i that appear in at least T% of the documents of the cluster.

T is a threshold value. In Table 3, T was chosen equal to 51%, and the meaningfulness of the clustering, using blind reviewers, as shown in Section 4.

ii. with weights:

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

Experimentation

See Section 4 for both the protocol and the results of the clustering and labelling module.

3.6 Query module

a. Query Processing Algorithm

THESUS can be effectively used for answering keyword based queries. In order to provide answers to such a query, we perform the following actions:

Map the keywords of the query to the ontology

Using Wu & Palmer similarity, we 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 only keep categories that have a weight above a certain threshold.

Find which cluster(s) is(are) closest to the categories that now define our query.

We calculate the similarity of our query, that is now a set of weighted categories of the ontology with all the labels of the clusters. Since the Wu & Palmer similarity between categories of the ontology is cached, this operation is very fast (approximately 5 μs to compute the similarity of a set of 5 categories with one subset. Thus under 1ms to compute the similarity with all the subsets in our example) This prunes out a lot of clusters to make the next steps faster.

Calculate the similarity between the keyword query, and every document in the selected clusters.

To do this, we consider their keyword description, since it is more precise. Once again, we will use here the similarity measure between sets of (weighted) words, this time the Wu & Palmer similarity is not cached (we have not precalculated all the similarities of words in WordNet) so this computation takes longer than the previous step, mainly due to the accessing of WordNet. However, since the number of documents in a cluster is on average a few hundreds, this step takes under a second in all our tests. We do not have the space here to explain the subtleties used to optimise this operation.

Return the 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.

b. Experimentation:

See Section 4 for details on the experimental protocol and for results.

4 Experimental evaluation

Our system produces results that can only be judged subjectively, since the relevance of a page or of the label of a cluster to the user who asked the query can only be answered by the user himself. The only reasonable way of evaluating our system is to use blind testers.

4.1 Clustering and Labelling

A tester is presented with the labels (i.e., sets of categories) of the clusters that our system has constructed. For a given set of documents (about 50 per tester) we ask the tester to classify the document into one of the categories, or into a 'noise' pile, if (s)he thinks that the document can not reasonably be labeled with any of those proposed. We indicate in Table 3 the percentage of documents that the human testers and THESUS classified into the same category, with various parameters of our clustering algorithm. With this method, we test the ability of THESUS to correctly name the clusters it discovers. We did not devise an experiment to see how human users would classify the sets of documents themselves, but let us note that THESUS never grouped more than 10% of the documents into the 'noise' pile, and usually only 5% of the documents remained unmapped to a cluster once THESUS had finished processing. This low value of noise, coupled with the high correlation between human testers, and THESUS clustering leads us to claim the clustering and labeling algorithm performs correctly.

MinSim	0.6	0.7	0.8	0.9
Correlation tester/THESUS	75%	82%	80%	68%

Table 3. Labeling algorithm results

The results are very satisfactory, and justify both our clustering and labelling heuristics. The drop in performance when *MinSim* is set too high is explainable by the fact our algorithm starts missing some clusters. The processing time for the clustering of 39000 documents took 45 seconds on a Pentium III, 450Mhz.

4.2 Queries

A tester defines a query comprised of a set of keywords. We submit this query to THESUS, and also submit it to popular search engines, without showing the results to the tester. We selected the top 5 results from THESUS, Google, Lycos, and Alta-Vista [Goo, Lyc, Alt] (We did not query Yahoo! Since it uses Google's results). We group these 20 pages and shuffle them to eliminate the ordering bias when we present them to the testers. For each page we keep a secret ID that identifies which system produced it. We show the 20 resulting pages to *all* the testers, and for each page, each tester gives a relevance mark ranging from 1 (irrelevant) to 5 (very relevant). We believe that this method yields completely unbiased results. We tested a broad range of queries about topics related to technology.

In the following table, we present, ordered by relevance, the percentage of documents returned by a selection of search engines and by THESUS.

Value	**THESUS**	**Google**	**A-V**	**Lycos**
V. Bad(1)	2%	6%	62%	40%
Bad (2)	6%	11%	10%	23%
Medium(3)	19%	33%	7%	15%
Good (4)	26%	31%	13%	13%
V. Good(5)	47%	19%	8%	9%

Table 4. Query results

These results show that THESUS returns significantly better answers to somewhat specialized queries than today's best search engines, and support our claim in favor of simultaneous use of incoming link semantics extraction, and clustering. The average time to answer a query was 1.4 seconds on a Pentium III, 450.

5 Conclusion

In this article, we presented THESUS, a system to enrich and classify documents from the web. Our results show that the use of a thesaurus and an ontology, coupled with the use of a novel similarity measure return relevant pages when queried in a manner similar to current search engines. Results to keyword queries are answered within seconds. Scalability is assured thanks to the use of clusters to eliminate a high number of documents irrelevant to the query.

Further work

We plan on further improving the clustering module, by testing other clustering algorithms. We also plan on testing other similarity measures to see if the degree of relevant results can be increased.

The greatest drawback in our system is that the thesaurus sometimes maps words to unrelated categories. We are currently investigating techniques to improve this.

6 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, 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).
We would especially like to thank our blind testers, Stratis Valavanis, Mathieu Daeron, Julien Heuvingh, Sandrine Launey and Antoinette Hubert.

REFERENCES

[ABF+98] R. Al-Halami, R. Berwick, et. al.Christiane Fellbaum editor, "*WordNet, an electronic lexical database",* Bradford Books. May 1998, ISBN 0-262-06197-X

[Alt] http://www.alta-vista.com/

[BP98] S. Brin, L. Page, "*The Anatomy of a Large Scale Hypertextual Web Search Engine"*, in the proceedings of WWW7 Conference.

[BK+90] N. Beckmann, H.P. Kriegel, R. Schneider, B. Seeger: *"The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles". SIGMOD Conference 1990.*

[BSF02] A. Bidault, B.Safar, Ch. Froidevaux, *"Proximite entre requetes dans un contexte mediateur"*, RFIA-2002, p653-662

[Cop] http://www.copernic.com/

[CD+98] S. Chakrabarti, B. Dom, D. Gibson, J. Keinberg, P. Raghavan, S. Rajagopalan, *"Automatic Resource list Compilation by Analyzing Hyperlink Structure and Associated Text"*, Proceedings of the 7th International World Wide Web Conference 1998

[CH79] W.B. Croft and D.J. Harper (1979) *"Using probabilistic models of document retrieval without relevance information"*, Journal of Documentation, 35(4):285-295

[Dmo] http://www.dmoz.org /

[EK+96] M. Ester, H-P. Kriegel, J. Sander, X. Xu, *"A density based algorithm for discovering clusters in large spatial databases with noise*", Proceedings of the 2nd International conference on Knowledge Discovery and Data Mining, 1996

[EM97] Thomas Eiter, Heikki Mannila, *"Distance measures for point sets and their computation"*, in Acta Informatica Journal, 34, 109–133 (1997)

[GGK01] A. Gionis, D. Gunopulos and N. Koudras, *"Efficient and Tunable Similar Set Retrieval"*, in proceedings of ACM-SIGMOD 2001

[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)

[Goo] http://www.google.com

[GRS98] Sudipto Guha, Rajeev Rastogi, Kyueseok Shim. *"CURE: An Efficient Clustering Algorithm for Large Databases"*, Published in the Proceedings of the ACM SIGMOD Conference, 1998.

[Gua98] Guarino N. *"Formal Ontology and Information Systems"* In N. Guarino (ed.), Formal Ontology in Information Systems. Proc. of the 1st International Conference, Trento, Italy, 6-8 June 1998. IOS Press

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

[Hen01] M. Henzinger, *"Hyperlink Analysis for the Web"*, IEEE Internet Computing, Jan.Feb.2001

[HGI98] Haveliwala, Gionis, Indyk, *"Scalable techniques for clustering the Web"*, in proceedings of Webdb98 worshop

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

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

[Lin98] D. Lin, *"An Information-Theoretic Definition of Similarity",* in proceedings of 15th ICML (1998)

[Lyc] http://www.lycos.com/

[Nii87] I. Niiniluoto, *"Truthlikeness"*, D. Reidel Pub. Comp., Dordrecht, Holland 1987

[Nor] http://www.northernlight.com

[PW00] T. Phelps, R. Wilensky, *"Robust Hyperlinks Cost Just Five Words Each"*, UC Berkeley Computer Science Technical Report UCB//CSD-00-1091. Berkeley, CA. 2000

[Res95] P. Resnik, *"Using Information Content to Evaluate Semantic Similarity in a Taxonomy"*, Proceedings of IJCAI-95, pages 448-453

[Rij79] C.J. van Rijsbergen, *"Information Retrieval"*, Butterworths, London

[RS-J76] S.E. Robertson and K. Sparck-Jones, *"Relevance weighting of search terms"*, Journal of American Society for Information Science, 27:129-146

[RSM] R. Richardson, A.F. Smeaton and J. Murphy *"Using WordNet as a Knowledge Base for Measuring Semantic Similarity between Words"*, Working Paper

[Sal87] G. Salton. *"Term Weighting Approaches in Automatic Text Retrieval"*, Dept. of Computer Science. Cornell University, 1987

[SMc83] Salton G., McGill M. *"Introduction to Modern Information Retrieval"*, McGraw-Hill, 1983.

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

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

[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