CLUSTERING WEB PAGE SEARCH RESULTS: A FULL TEXT BASED APPROACH

Ricardo Campos [1, 2], Gaël Dias [1], Célia Nunes [1], Bono Nonchev [1]

[1] Centre for Human Language Technology and Bioinformatics, University of Beira Interior,
Rua Marques d´Ávila e Bolama, 6200-001 Covilhã, Portugal
{ricardo, ddg, celia, bono}@hultig.di.ubi.pt

[2] Polytechnic Institute of Tomar, Quinta do Contador, 2300-317 Tomar, Portugal,
ricardo.campos@ipt.pt

Abstract

With so much information available on the web, looking for relevant documents on the Internet has become a tough task. In this paper we present as approach which is a match between a query-based Google and a category-based Yahoo. WISE is a web page hierarchical clustering system, language and topic independent, supported by a graph based overlapping algorithm that groups web relevant pages into a hierarchy of properly labeled overlapping clusters. We consider phrases instead of single words, web content mining techniques to represent the documents, and an hierarchical soft clustering approach as opposed to a flat clustering one. Together, this will improve the organization of the results, enhancing the experience of looking for relevant and hide documents and helping on solving the synonym and ambiguity problem.

Key Words: search engines; hierarchical web page clustering; web information search and retrieval, web mining; web intelligence;

1. Introduction

Current search engines return lists of ranked urls with their title and their snippet (a short description of the document), but still fail to find relevant contents and to present them in an organized way. In this process, the user is required to go through the extensive list of the retrieved results, before concluding that the relevance defined by the search engines, does not satisfy its needs.

Whilst traditional search engines continue to present data in a linear fashion, some commercial approaches, like Vivisimo (*www.vivisimo.com*), iBoogie (*www.iboogie.com*), Clusty (*clusty.com*) and Grokker (*www.grokker.com*), began to present it in a partitioning hierarchical one, an innovative approach in information retrieval, that helps users in the process of seeking for relevant web pages, otherwise undiscovered due to their location in the ranking list. Known as post-retrieval document browsing or ephemeral clustering, this process doesn't even require the use of pre-defined categories, like in classification methods [1], what makes it to be a bottom-up process with the categories being part of the output rather than part of the input [2].

In this paper, we present a meta-search engine called WISE that builds soft hierarchical clusters on the fly, without pre-defined groups or pre-built knowledge bases, by applying an overlapping clustering algorithm called PoBOC (see [6]), which is graph based and allows a document to be in multiple clusters (overlap), reflecting the fact that a web page may contain different meanings of the query terms. To represent the contents of the web documents, we use web content mining techniques, introduced in the context of the Webspy software (see [3]), and statistical methodologies for phrase detection, with the use of SENTA software (see [4], [5]).

The system, instead using web snippets, fully analyzes web pages in its entirety, increasing its recall. Web page clustering is a challenging variant of web snippet clustering, but as [7] refer, there is a closer relationship between the analysis of the overall web page and the quality of the final results, increasing the likelihood of retrieving relevant phrases and quality hierarchical clusters, as more information is considered in the analysis. By doing this, we also keep independent of the quality of the web snippets retrieved, regardless the particularities of each search engine.

Overall, we think our solution is innovative, as the system shows interesting ability to find, analyze and understand the content of the web pages, disambiguate the query and organize all the information retrieved by any search engine in response to a query language and topic independent. A good illustrative example is the following: suppose we perform the query *Benfica* in the system. So, instead of presenting a list of results, it will present two main clusters: one related with the well-known Portuguese football team, and the other one related to the neighborhood of Lisbon named *Benfica*. We will see this in detail in the case study result.

Following, we provide a survey of the current literature and the state-of-the-art in the field. Then we explain our main contributions defining the architecture and its implementation, to finally show some case study results, detail conclusions and propose some ideas for future work.

2. Related Work

In this section, we provide a survey of all different approaches proposed so far. Table 1 summarizes all these works.

Table 1. Different approaches to web snippet clustering.

	Flat Clustering	Hierarchical Clustering
Single Words	[14], [19]	[8], [15]
Phrases	[1], [9], [13]	[2], [7], [11], [12], [16], [17], [18]

The first step of any of these systems is to select a set of keywords (also known as keys or topic concepts) to summarize and describe each document. [7], [8], [9] rely on choosing the k most significant words of the document based on term frequency and inverse document frequency (also called TF.IDF [10]), whilst [1], [11], [12], [13], [14], use shared n-grams between the web snippets. [15]use concept lattice to return as relevant terms the neighbors of the focus concept by taking advantage of word proximity. [16] adopt a hybrid solution by considering the vector space model and shared n-grams representation. [17] use topicality and predictive properties to find relevant terms. [2] consider key topics as pairs of words linked by lexical affinity and

finally, [18] select as document features, noun phrases and single words that are single nouns or adjectives. The different approaches also distinguish themselves, as shown in Table 1, by the match of the following two items:

(1) single words or phrases and
(2) flat or hierarchical clustering

The simplest case is the match between flat clustering (only one-level partitioning of the data) and single words, whilst the general case is the match between hierarchical clustering and phrases. In the simplest case using a k-means algorithm is Scatter/Gather [19], one of the firsts works to appear, although it has never been tested on web environment. Another one is Retriever [14], which clusters the results returned from a search engine based on a relational fuzzy clustering algorithm (RFCMdd) and the identification of k-medoids. Mooter (*www.mooter.com*), a commercial search engine, also belongs to this category.

Grouper, developed by [13], is the first to do flat clustering with phrases. They propose an algorithm named Suffix Tree Clustering, where each node of the tree is a phrase and a base cluster (documents sharing a phrase). [1] extract phrases as candidate cluster names based on a regression model over five different features and the clusters are generated by merging the documents that share the same phrases with a given threshold. Lingo, developed by [9], uses Single Value Decomposition (SVD) over the document distribution of words and phrases to perform the same task.

FIHC, developed by [8], provides hierarchical clustering but with single words. The clustering process is based on the idea of frequent itemsets which are a set of words that co-occur in some minimum fraction of documents in a given cluster. CREDO (*credo.fub.it*), developed by [15], uses single words and the concept of lattice to produce hierarchical clustering.

The match between hierarchical clustering and phrases seems to be the best approach as [7] show improved results compared to others. SnakeT, developed by [7], uses a snippet analyzer

extracting phrases such as named entities by applying part of speech tagging and using a simple threshold to evaluate if a phrase appears in other documents. The hierarchy construction process consists, like many other works described here, in the following: documents sharing the same sentence belong to the same cluster. DisCover, developed by [18], tries to discover the maximum number of documents that can be described or covered by a set of features which are generally noun phrases and single words that are single nouns or adjectives. SHOC, developed by [12], uses an algorithm based on suffix arrays for key phrase discovery and uses SVD to reduce data sparseness before applying hierarchical clustering. Tumba, developed by [11], considers phrases as groups of two words co-occurring and uses the STC clustering algorithm (developed by [13]) and the subsumption algorithm (developed by [20]) to produce the hierarchy of concepts. MSEEC, developed by [16], discovers phrases by using a clustering algorithm based on the LZW compression method and produces hierarchical clusters with the modified theme detection method which is basically a hybrid solution as it considers a theme candidate to be a cluster based on the frequency of its concurrency and if it is contained in more than one document. The system called CIIRarchies, developed by [17], uses a probabilistic technique to find phrases combined with a threshold and a pre-computed language model which combines the use of two properties (topicality and predictiveness) to find topical terms in a document and builds a hierarchy using a recursive algorithm. LAH (Lexical Affinities Hierarchical clustering) developed by [2], considers key topics as pairs of words linked by a lexical affinity where LA stands for a correlation of their common appearance combined with an optimal agglomerative complete-link hierarchical clustering algorithm. We can also find in this category the following commercial search engines although their technology is not public available: Vivisimo, iBoogie, Clusty and Grokker.

It is important to notice that all these methods share the use of titles, stop words and stemming algorithms. If in one hand their solution becomes faster, on the other hand, it becomes language-dependent, hindering their adaptation to real-world environment. The systems use web snippets and only two works try to raise their knowledge by adding more information: [7], the most up-to-date system in the field, enriches web snippets by adding two existing knowledge bases previously compiled and thus static, whilst [1] considers the use of several features to extract key concepts from snippets, something similar to our approach, but only based on flat clustering.

Overall, the works published, have ignored the potential of using web content mining techniques to semantically analyze web pages, hindering the full understanding of web documents. As a consequence, the ambiguity problem (a query term may have more than one meaning) and the synonymy problem (web documents may only have just synonyms of the query term), which are central issues in the context of modern IR that tend to get worse when the user is not familiar with the topic he is looking for, still remain unsolved. In addition, the systems are language dependent once they use stop words and stemming algorithms, and most part of the clustering algorithms used, require the user to specify the number of clusters as an input parameter, and use thresholds to evaluate if a document belongs to a cluster or not.

To deal with these drawbacks, we propose a hierarchical soft clustering phrases approach. For that purpose we use web content mining techniques to extract key concepts from documents, combined with a statistical phrase extractor named SENTA and soft clustering algorithm called PoBOC, both parameter free. In the following section we fully detail its architecture.

3. The Architecture of Wise

WISE is a web search system. Unlike all the methodologies proposed so far in the literature, our full text based analysis system, doesn't

considers the use of stopwords, stemming algorithms nor thresholds, being completely language and topic independent. Its architecture is modular, and each part of it, can easily be used individually by other researchers.

In response to a query (the user can choose which search engine will run it), the system returns a page with a set of clusters and their associated key concepts which are keywords representing the web documents, within which is a list of urls that can belong to more than one cluster, so that the user can easily choose the web page he wants to see. Our algorithm is composed of five steps:

1. Search results gathering;
2. Selection of relevant web pages;
3. Document parsing for phrase extraction;
4. Document parsing for key concept extraction;
5. Hierarchical clustering and labeling.

As a result, we propose a structured catalogue of retrieved urls instead of an ordered list of relevant documents. The overall architecture is formalized below, based on the five steps mentioned above.

The first step is formalized in Equation 1 where q is the query, d_i a document, r_s a function of a search engine s, which computes the probability of relevance between q and d_i so that R is the set of the retrieved n documents by the search engine for the query.

$$R = r_s(d_i \mid q), \forall_i, i = 1,...,n. \quad (1)$$

The set of the retrieved documents, are nonetheless treated as if they had equal relevance, disregarding the fact that the estimated relevance of a document decreases as more results are gathered with many of the urls returned having little or no connection with the query [14], probably decreasing the quality of the final cluster results. For that reason, we developed a new extraction method of relevant web pages that ignores some of the retrieved documents and adds some others.

3.1. Selection of Relevant Web Pages

So, in the second step, we apply a selection function over R as defined in Equation 2:

$$R' = select(R). \quad (2)$$

where *select* is the algorithm that selects the most relevant documents over the set of web pages search results, by selecting:

(1) any absolute url retrieved (i.e. the web domain) and
(2) any relative url which number of occurrences (i.e. the number of times that for each retrieved url, its web domain appears in the list of the results) exceeds the *average_relevance* value, defined in Equation 3, where "# different absolute urls" is the number of distinct absolute retrieved urls (i.e. web domains).

$$average\_relevance = \frac{R}{\# \ different \ absolute \ urls}. \quad (3)$$

In order to better understand our procedure, consider Figure 1, where the average relevance takes the value 4/3 (the equivalent to 1.33333). So our system will select, regardless its number of occurrences, all the absolute URLs retrieved (in this case, www.vodafone.com and www.manutd.com) and disregard all the relative URLs whose number of occurrences is below the calculated average relevance (in this case the website http://geocities/mobiles/test.html).

WebSite	Nº Occurences
www.vodafone.com	2
www.vodafone.com/news/01.html	2
geocities/mobiles/test.html	1
www.manutd.com	1

Figure 1: Selection of Relevant WebPages.

Additionally, considering that some web pages are not properly indexed and consequently not captured, although probably related to the query, the algorithm extends the list of web pages previously selected (see Figure 2). For that purpose, for each absolute url retrieved, we rerun again the search engine over the same query, in order to catch its n best pages (being n a fixed value).

Figure 2: Catch the *n* best pages of each absolute URL.

Next we parse the documents for phrase extraction. This would benefit the work of search engines in the process of classification, by returning more intuitive key concepts and cluster titles. Nevertheless, disregarding the use of phrases is a loss of valuable information [13], most traditional search engines keep treating each document as a bag of single words ignoring important contextual information that would increment the knowledge obtained from the documents as it is shown in [21], [22], [23], [24], [25], [26] and [27].

3.2. Document Parsing for Phrase Extraction

So in this third step we use SENTA [4], [28], a statistical methodology that uses three basic concepts: positional n-grams (ordered vector of words), the association measure Mutual Expectation (ME) [28] and the GenLocalMaxs algorithm [29]. SENTA is based on a well-founded mathematical model and has been implemented using suffix arrays and the Multikey Quicksort algorithm [5] to be used in real-world applications. The ME is based on the concept of Normalized Expectation (NE), whose basic idea is to evaluate the cost, in terms of cohesion, of loosing a word in a positional n-gram. Thus, the more a set of words is joined, the less it accepts the loss of any of its components and higher its NE value should be. Also, knowing that one of the most relevant criterions to identify phrases is their frequency [30], [31], the ME of any positional n-gram is defined as the product of its NE and its relative frequency. Then, the GenLocalMaxs algorithm searches for positional n-grams which ME value is a local maxima and define them as phrases. These multiword units then replace the single words in the set of relevant web pages.

This step is defined in Equation 4, where each document within *R'* is parsed by SENTA to extract phrases, which subsequently replace the relevant sequences of single words, giving rise to a new set of documents known as *S*:

$$S = senta(d_j), \forall d_j \in R', j = 1, ..., \#R'. \quad (4)$$

Each phrase (from now on we indistinctively refer to phrases, as terms in the documents i.e. single words or multiword) of the document is represented by a set of eight properties calculated during the next step.

3.3. Document Parsing for Key Concept Extraction

As [15] refer, search engines suffer from a lack of a concise representation of the retrieved documents. [17] consider that the main challenge in creating hierarchical clusters is to select terms that will accurately describe the documents. Instead of using vector space model, shared n-grams, etc.., our system uses web content mining techniques to extract web knowledge (possibly hidden) by analyzing the content of the documents [32], relying on the definition of eight features (below introduced), divided into two categories: the first six characterizing the importance of the phrase in the document and the following two evaluating the relationship between the phrase and the query in the document collection.

In the following we present the eight properties, representing the current phrase as *x,* the documents d_j, *j=1,…,n,* as being all the documents where *x* occurs, $\#d_j$ as being the total number of phrases in the document d_j, and the query as *q*. All the features that depend on the number of documents are normalized to give a value between [0;1].

Inverse Document Frequency
This feature was proposed by [10] and represents the dispersion of a given *x* in the set of documents where *x* appears i.e. d_j, *j=1,…,n.* Equation 5 gives the non normalized equation.

$$IDF(x) = \log_2 \frac{\#R'}{n}. \quad (5)$$

The *IDF* measure retrieves a value between [0;+∞[. However, to be size-independent, we need to normalize this measure, so that it returns

a value between [0;1]. This is done by introducing Equation 6.

$$IDF'(x) = \frac{IDF(x)}{\log_2 \#R'}. \quad (6)$$

Normalized Text Frequency
This feature, given by Equation 7, is the average frequency of each word in the set of documents where $numOcur(x|d_j)$ is the number of occurrences of x in the document d_j.

$$TF(x) = \frac{\sum_{j=1}^{n} \frac{numOcur(x \mid d_j)}{\#d_j}}{n} \quad (7)$$

Normalized Density
The normalized density is based on the same idea as the rank analysis used in automatic global analysis [33]. It is given by Equation 8 and 9, and is the average of a feature named AD (Average Distance), calculated over all the documents where x appears. Generally, we define AD as the normalized distance between all the occurrences of x in the document d_i as follows where $dist$ is a function that calculates the distance between consecutive occurrences of x. If x appears just one time in the document dj, then $AD(x,d_j) = 0$.

$$AD(x,d_j) = \frac{\sum_{j=1}^{numOccur(x|d_j)} \frac{1}{dist(x_{j+1}, x_j)}}{\#d_j}, j = 1,...,n. \quad (8)$$

Consider Figure 3 where $AD(x,d_j) = \frac{1}{10} + \frac{1}{5}$
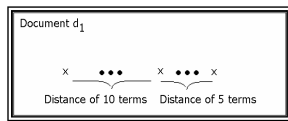

Figure 3: Distance between all the occurrences of x in doc d1.

The normalized density *ND* is then defined as follows in Equation 9.

$$ND(x) = \frac{\sum_{j=1}^{n} AD(x,d_j)}{n} \quad (9)$$

Normalized First Position
As in [34], we use the normalized first position feature (terms at the beginning of the document tend to be more important), given by Equation 10 where $fp(x|d_j)$ is a function that counts the number of phrases behind the first occurrence of x in the document d_j, i.e., if x occurs in a document, the system considers its first occurrence, counting the number of terms behind it and calculating its average over all the documents considered.

$$FP = \frac{\sum_{j=1}^{n} \frac{fp(x \mid d_j)}{\#d_j}}{n} \quad (10)$$

Size in Characters
The feature size is simply the count of characters of x and retrieves a value between $[0;size(x)]$.

Capital Letters
The feature represents the number of capital letters of x which is between $[0;capital(x)]$.

Equivalence Index
The Equivalence Index [35] is a co-occurrence measure which evaluates the strength existing between x and q. This measure is defined in Equation 11 where $f(x,q)$ is the absolute frequency of x and q occur simultaneously in a window of n characters and $f(x)$ and $f(q)$ are respectively the absolute frequencies of x and q.

$$EI(x,q) = \frac{f^2(x,q)}{f(x).f(q)}. \quad (11)$$

Considerer the following example, where d1, d2, d3, d4 and d5 are five documents retrieved by the search engine, x the term and q the query. According to Figure 4 $f(x,q)=5$, $f(x)=6$, $f(q)=7$, being $EI(x,q) = \frac{5^2}{6x7}$.
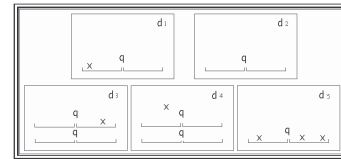

Figure 4: Strength existent between x and q.

Normalized Distance to the Query
This feature, given by Equation 12 and 13, is the average of a feature named *SD* (Shorter Distance), calculated over all the documents where x appears. Generally, we define *SD* as the shortest normalized distance between x and q in the document d_i as follows where *fdist* is a function that calculates the shortest distance

between the occurrences of x and q in the document d_i.

$$SD(x,q,d_j) = \frac{fdist(x,q,d_j)}{\#d_j}, j = 1,...,n, \quad (12)$$

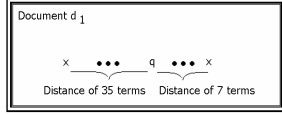According to Figure 5, SD (x,q,d1) would be 7.

Figure 5: Shortest distance between x and q in document d1.

The normalized distance to the query is then defined as follows in Equation 13.

$$DQ(x,q) = \frac{\sum_{j=1}^{n} SD(x,q,d_j)}{n} \quad (13)$$

To combine these eight properties into a single score, we apply, for each phrase of a document, a decision tree model (C5.0 class). It is clear, as many authors have shown [34], that decision trees apply particularly well to textual data. As [1] refer, although a supervised learning method requires additional training data, it makes the performance of search result grouping significantly improved (particular tests were made with linear regression and neural networks, leading to inferior results). The decision tree was built over 5 pruned decision trees previously trained using a 5-fold cross validation and applying over-sampling to avoid data sparseness. The model was trained over a set of newspapers article gathering three distinct domains (sports, politics and society) in order to assure its generality, domain and topic independence, obtaining 82.49% of average precision over the five test data to classify positive and negative examples, and 60.72% average precision over the same five test data to classify only positive examples.

As a result of the webspy decision tree, each document d_j of S, is therefore represented as a vector of key concepts $kc_j. = (kc_{j1}, kc_{j2,...,} kc_{jm})$, $j=1,...,\#R'$ where m (whose value can be different for different documents) is the number of key concepts for d_j retrieved by the Webspy software [3] taking into consideration the query q as formalized in Equation 14

$$kc_j. = webspy(d_j \mid q), d_j \in S. \quad (14)$$

The higher the score retrieved by the decision tree (see Figure 6), the more relevant the key concept will be for the given url/document and the query.
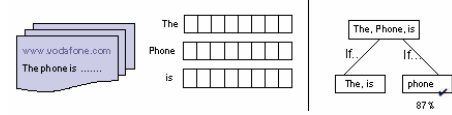

Figure 6: WebSpy selects, among many others, the word *phone* as a relevant key concept with 87%.

The representation of any document is then given by a vector of key concepts containing its most relevant scored phrases (see Figure 7).

Figure 7: Documents represented by a vector of key concepts.

Each element of the key concept vector, also known as key concept (kc), is a flat cluster with a list of related urls (see Figure 8).

Flat Cluster	WebSites
Manchester	www.vodafone.com
	www.manutd.com
Phone	www.vodafone.com
T1	www.era.com
Rent	www.era.com
Cristiano Ronaldo	www.manutd.com
Season Ticket Renewals	www.manutd.com

Figure 8: List of Flat Clusters.

Each key concept kc_h, $h = 1,...,u$, can then be defined as a cluster C_h, so that it contains all the documents d_j, where d_j is relevant through Webspy to both q and kc_h, as given by Equation 15 where dt is the decision tree model that is implemented in Webspy and defines if a key concept is relevant to any document. Following, we define the list of flat clusters such as $FC = \{C_h \mid h = 1,...,u\}$.

$$C_h = \{d_j \mid d_j \in S \text{ and } dt(d_j, kc_h)\}. \quad (15)$$

Although being an important step in the organization of the results, flat clustering still doesn't solve the ambiguity problem as the set of results are displayed in an unorganized partitioning way, sharing different concepts of the query (see Figure 8).

As [17] refer, a user may look at several pages of results from a ranked list before finally concluding that there is nothing relevant. So beyond solving ambiguity and synonym problem, the introduction of hierarchy will help the user to determine in a glance what is relevant and what is not. So we propose in the following section a soft clustering hierarchical structure.

3.4. Hierarchical Soft Clustering and Labelling

In order to prepare the clustering step, the WebSpy software, considers each flat cluster kc_h as a query, running it, over all webpages belonging to the flat cluster.
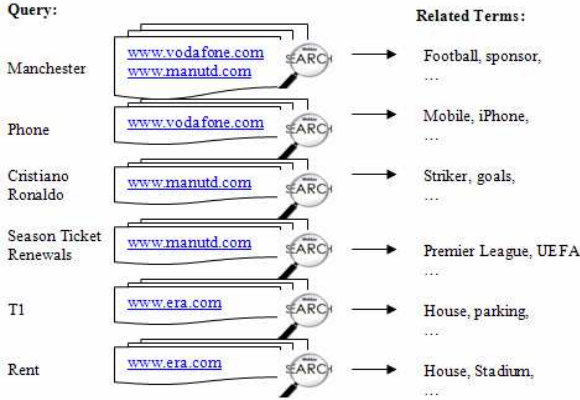

Figure 9: List of Flat Clusters.

In result, each flat cluster is also represented by a set of related phrases with a given relevance probability (see Figure 9)

To formalize, we represent each document d_v, $v=1,...,\#C_h$, of each flat cluster C_h, as a vector of key concepts $kc_{v.} = (kc_{v1}, kc_{v2,...,} kc_{vt})$ where t is the number of key concepts for d_v retrieved by Webspy taking into consideration the query kc_h.

$$kc_{v.} = webspy(d_v \mid kc_h), d_v \in C_h. \qquad (16)$$

Then, we apply the PoBOC algorithm, to obtain the hierarchy of flat clusters C_h.

$$HC = poboc(\{sim(kc_{v.}, kc_{v'.}), v \neq v' \wedge v, v' = 1,...,\#C_h\}). (17)$$

using the cosine similarity measure,

$$\cos(kc_{v.}, kc_{v'.}) = \frac{kc_{v.}.kc_{v'.}}{|kc_{v.}|.|kc_{v'.}|}. \qquad (18)$$

defined in Equation 17 as the *sim* function, in order to build the similarity matrix, by applying it to all flat clusters

Figure 10: Similarity Matrix.

By applying the PoBOC algorithm to the flat clusters, we propose a disambiguation methodology, as key concepts with different meanings should be gathered in different clusters. By doing so, the possible different meanings of the query should be evidenced. In fact, like [1] and [7] refers, the organization of the results into clusters eases the user's navigation when compared to the navigation through an extensive list of results. Some clustering algorithms, however, are not prepared for web clustering as they do not satisfy the requirements for clustering documents defined by [8]: high volume of data, easy for browsing and meaningful cluster labels (a comparison between clustering algorithms can be found in [1], [8], [36], [37], [38]).

PoBOC, in its turn, and unlike the k-means or the EM (just to cite those), can be used on the fly as it does not depend on any input parameter (e.g. k clusters and/or thresholds). Moreover it is based on graph theory and has shown encouraging results [6] compared to other clustering algorithms when applied to textual data, building a hierarchy of concepts, based on the following four main steps:

(1) construction of the similarity graph, based on the similarity matrix and local maxima of similarity values. (2) the search of poles in the similarity graph i.e. sub-graphs that are clique graphs (finding a clique graph is a NP Problem. As a consequence, [6] proposed a heuristics that allows to approximate maximum-size clique-graphs in order to run the algorithm in reasonable time. The heuristics selects a number of vertices and a notion of neighborhood to calculate the clique-graphs). (3) the evaluation of membership of each flat cluster key concept to all the poles, (4) the assignment of the flat cluster key concept to one or several poles.

With this clustering step (see Figure 11), we aim at disambiguating the sense of the query term, helping the user in his search:
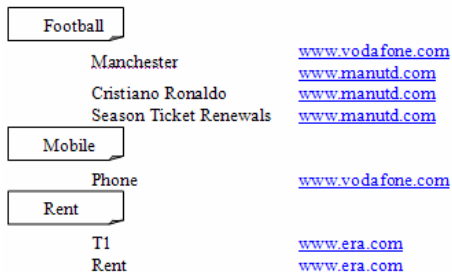

Figure 11: List of Hierarchical Clusters.

Finally, each cluster is labeled (see previous figure) using a simple heuristics that chooses the key concept that occurs more often in the vectors of each cluster key concept, taking into account the sum of scores in case of ties.

$$HC' = label(HC). \qquad (19).$$

4. Implementation of WISE

Often, web servers are slow or not responsive, mainly due to the significant latency of the network but also to the processing time. In the case of a meta-search engine this gain even more importance, as the computation of measures cannot start before all texts are downloaded from the Internet. Additionally, whilst a normal snippet is about 200-300 characters, in our architecture we do not consider texts smaller than 1000 characters (on average the processed texts are 10KB of size), which turns it, even more computationally intensive.

To be real-world adaptable and decrease the responding time due to our full text based approach, we propose a distributed system to handle all queries. Large scale applications in science rely on larger configurations of parallel computers, often comprising hundreds of processors. There are, however, several problems that an implementation of a program has to solve, or more accurately, balance. As we have already mentioned speed is a major problem. In our case, we are computing metrics over the whole text which takes on average 10 seconds on a celeron-like CPU. With an usual amount of 30-40 texts per query it would take for a single CPU more than 40-50 minutes to answer to the user query. This is obviously unacceptable, so we took advantage of distributed computing network, splitting the program into independent modules that can be hosted on any computer connected to one another via RMI, which is a network transparent mechanism. A diagram with all the different modules and their connections can be seen in Figure 12.
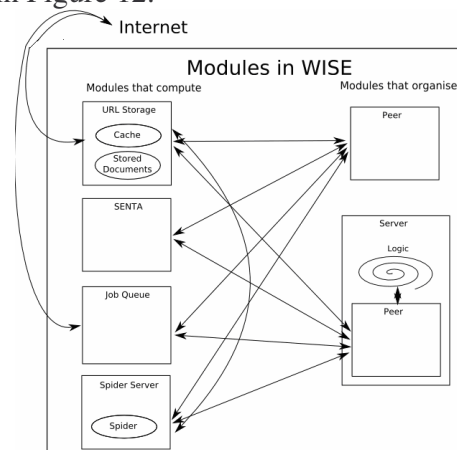

Figure 12: Modules in WISE and its connections.

As we can see, there are two types of modules, computing and organizing modules. We explain all of them in the following:

(1) **URL Storage**: This module does all the data connections to the Internet. If another module needs data from the Internet, it connects to this module with request. This approach has one main benefit. Since all data is downloaded by this module, we can check for malicious HTML and we can implement timeouts in a single place.

(2) **SENTA:** This module handles the requests to the software for finding phrases. When a text is retrieved from the Internet and all HTML tags are stripped out, we run SENTA and receive a list of phrases that will replace single words when appropriate.

(3) **Spider Server:** This module handles requests to the spider called Webspy. In WISE, Webspy is the module that computes the measures over the documents to extract their keywords. It connects to the Internet to download files. In particular, we modified the spider to connect to the URL Storage to retrieve the files.

(4) **Job Queue:** This module, which is just the direct output of the program, receives queries from different clients to WISE, makes the computations and returns the results. There are also two modules (peer and server) that are used to organize the process of the computations. They are independently addressable, and if we want to use a Spider Server, we have to create a peer or a server to host it.

(5) **Peer:** This module represents a single computer (or peer) in a network. It is identified by the host name of the computer it is on, and the port it is listening to.

(6) **Server:** This module is the heart of the network. As long as the server is on-line, the network will be working. If one or more peers are removed, the server will detect they are no longer responsive, and will send the remaining tasks to other computers. If the server is down, all currently working jobs will be lost. In the server there is a class commanding the whole network, which makes the distribution of the tasks, adding new peers and checking whether peers are still connected.

Naturally, some combinations of modules work better than others, but this depends on the network configuration. In the testing process, we found that the best configuration for our network, which had one bi-processor server and several celeron-like PCs with fast access to the Internet, was the one shown in Figure 13.
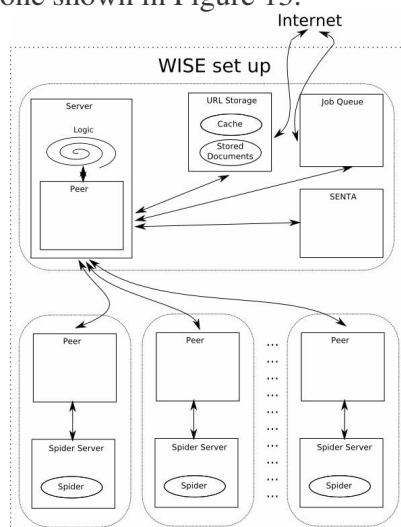

Figure 13. Distribution of the modules.

First, we noticed that the web spider takes up almost all of the processing time. We did not have problems with the Internet connection, so we put the URL Storage only on the server. It is possible to put it on many computers, to speed up download, but that is not needed at this stage.

Second, since the spider uses most of the processing time, we put it on all clients. It is slow for one computer, as we mentioned, but is working well enough with 5 computers when considering 20-30 results from the search engine, which we deem are enough for a good result.

Third, we put the server on the fastest computer, because when the tasks to the spiders are being distributed there is a small amount of processing time that is used by the computer hosting the distributing tasks.

Finally, the last thing to note is that the fastest computer in our network, which has two cores with HyperThreading, is running not only the server, but also four clients. That is done in order to create enough loads for it. To better understand the path of the logic through our system, consider the following steps illustrated in Figure 14:

(1)   The query comes from a client on the Internet. For example, the query is something like: I want all results for the query *Manchester*, while considering *20* results from the Google™ search Engine.

(2)   The results from Google™ are downloaded via the URL Storage.

(3)   The texts are parsed and then processed through SENTA to retrieve the phrases.

(4)   We save the resulting texts in the URL Storage. So later, the web spiders can retrieve the results from there instead of requesting over the network again. This step cannot be missed as we run the spider not on the original texts, but on the texts that have been parsed and passed through SENTA.

(5)   (6) and (7) Then, we send different texts to different web spider on different peers to retrieve all relevant terms/key concepts for each document. All this is done in parallel with all the peers working on their separate tasks simultaneously.

(8) After all relevant terms are found, the PoBOC algorithm is run to determine the clusters and finally, all clusters are labeled. All this is done by the main server as all results must be gathered to perform the soft clustering.

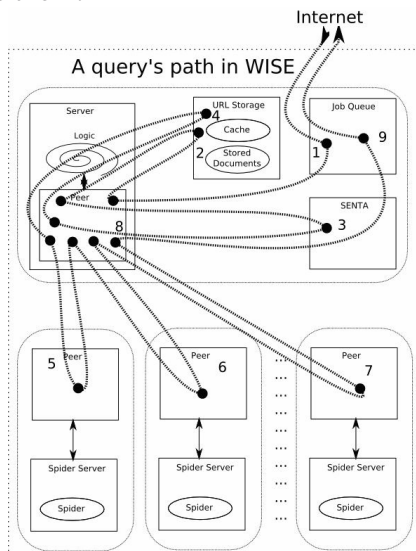(9) Finally, the results of the user's query are saved in the Job Queue module, so that the user receives them.

Figure 14. A simple query path through the system

Since the modules can be run independently, they are run in parallel whenever the results needed for the next computations are ready. This way the program actually benefits from using distributed processing.

As a result, by using a small network of only 3 computers, WISE is able to compute tasks about 10 times faster than when based on one single server. For instance, a result with 40 relevant results from Google™ is evaluated for 4 minutes, with nice and structured results returned. Although this processing time still seems to be overwhelming, if more peers are installed, the processing time will automatically decrease to reasonable time, however, there is still work to do in web spider parallelization in order to increase the overall speed. Indeed, as all features are independent from each other, it is easy to compute them in parallel and gather the final results on a main peer. This task is under development and should allow to compare WISE with other commercial meta-search engines which only use snippets in real-time access. This

will be a major advance as we will be able to compare snippet based techniques to our full text based technique.

5. Results

In this section, we present some case study results returned from WISE in response to the user's query *Benfica* on May 19th 2008, using the Google™ search engine. We intend to show some characteristics of our system such as: phrase and cluster identification, concept disambiguation and language-independence.

The cluster in Figure 15 refers to the set of urls related to *Michael-Laudrup* the ex-former football coach of *Getafe*, spoken at that time to be a possible successor of *José-António-Camacho,* who in turn was replaced in his first passage by *Benfica* for *Giovanni-Trapattoni*.

Figure 15. Cluster *José-António-Camacho*.

We can notice that the label shows some degree of quality and semantic description of the content of the cluster due to the identification of relevant phrases such as the named entity *Michael-Laudrup*. One other interesting issue is the capacity of the system to deal with word mistakes (*Giovanni*, not *Geovanni*). This result is due to the fact that no restriction is made over term frequency and/or document representation.

In Figure 16, we can see the ability of WISE to deal with concept disambiguation. As we have already seen, *Michael-Laudrup* is related to *Benfica* football club, but the system also retrieves a cluster labeled *PS* which refers to the socialist political party located in the *Benfica* neighborhood and another one labeled *Universitários* referring to student life like housing, transports and roads as *Benfica* is a privileged area for university student housing.

Figure 16. Cluster *PS* and Cluster *Universitários*.

Our last example in Figure 17 shows the language-independent characteristic of WISE as it gathers urls with different idioms, in this particular case, Portuguese, English and Hungarian.

Figure 17. Language Independence.

On one hand, the Hungarian link related to the well known Miklos Feher, suddenly died in the lawn when he was playing. On the other hand, *Benfica* football club and his performance in the European Champions League, resulting in many results in English.

6. Conclusion and Future Work

As [1] refer, it is clear that organizing web search results fastens the user browsing process as the task of searching for relevant web pages in an unstructured list of results is time consuming. Although, major search engines have not yet considered implementing clustering visualization and only a few commercial search engines are beginning to appear, *Vivisimo* being the most famous. This paper explores web page hierarchical soft clustering as an alternative method for organizing search results.

With WISE, we attempt to make search results easy to find through a list of well identified clusters with quality labels. Moreover, we provide a real-world web architecture as WISE is topic and language-independent, and allows query disambiguation and document overlapping. WISE has a strong theoretical basis, but its architecture is simple. In particular it is based on

(1) an algorithm ignoring less relevant documents and increasing relevant ones; (2) statistical parameter free phrase extraction to define concepts and thus enhancing document understanding; (3) web content mining techniques to represent document contents; (4) a soft clustering parameter free algorithm to organize results into a hierarchy of concepts, (5) a classical labeling process and (6) a distributed implementation.

The architecture and the proposed algorithms try to address one important problem in IR: returning quality results through an organized and disambiguated structure of concepts, but formal evaluation is still needed. We planned to test WISE against all available web snippet clustering search engines existent in Table 1 but none of the works referred in the literature is available online. Despite previous works in this area is very scarce [9], in the future, we plan to execute user feedback surveys, which have been the most favored techniques for estimating the algorithms usefulness [39], although its logistic difficulty and underlying subjectivity. We also plan to evaluate WISE within TREC by using the key concepts existent in each cluster to perform automatic query expansion and evaluate the quality of the retrieved results.

Finally, we are currently improving the performance of the system, to complete an on-line version (*http://wise.di.ubi.pt*) that will allow users to perform a query using any search engine. In addition, the system which is currently making all the computation on the fly, will integrate in the future a web warehouse to save the key concepts of the documents, incorporating user feedback in the loop, and avoid running the system whenever a query is performed, by providing a self-contained and auto-fed indexing structure.

References

[1]   H. Zeng, Q. He, Z. Chen and W. Ma, "Learning to Cluster Web Search Results", in *Proc. of the 27th Annual International Conference on Research and Development in Information Retrieval*, Sheffield, UK, 2004, pp. 210-217.

[2]   Y. Maarek, R. Fagin, I. Ben-Shaul, and D.

Pelleg, "Ephemeral document clustering for web applications", *Technical Report RJ 10186*, IBM Resarch, 2000.

[3] H. Veiga, S. Madeira, and G. Dias, "Webspy", *Technical Report nº 1/2004*. http://webspy.di.ubi.pt, 2004.

[4] G. Dias, S. Guilloré, C. Bassano, and G. Lopes, "Extraction Automatique d'unités Lexicales Complexes: Un Enjeu Fondamental pour la Recherche Documentaire", in *Traitement Automatique des Langues*, Vol 41, Paris, France, ISBN:2-7462-0225-5, 2000, pp.447-473

[5] A. Gil, and G. Dias, "Using Masks, Suffix Array-based Data Structures and Multidimensional Arrays to Compute Positional Ngram Statistics from Corpora", in *Proc. of the Workshop on Multiword Expressions of the 41st Annual Meeting of the ACM*, Sapporo, Japan, July 7-12, 2003.

[6] G. Cleuziou, L. Martin, and C. Vrain, "PoBOC: an Overlapping Clustering Algorithm. Application to Rule-Based Classification and Textual Data", in *Proc. of the 16ᵗʰ ECAI*, Valencia, Spain, 2003, pp. 440-444.

[7] P. Ferragina, and A. Gulli, "A Personalized Search Engine Based on Web-Snippet Hierarchical Clustering", *Software: Practice and Experience*, Vol 38, no. 2, pp. 189 – 225, 2008.

[8] B. Fung, K. Wang, and M. Ester, "Large Hierarchical Document Clustering using Frequent Itemsets", in *Proc. of the SIAM International Conference on Data Mining*, San Francisco, CA, May, 2003.

[9] S. Osinski, J. Stefanowski, and D. Weiss, "Lingo: Search results clustering algorithm based on Singular Value Decomposition", in *Proc. of Intelligent Information Systems Conference*, Zakopane, Poland, 2004.

[10] K. Spärk Jones, "A Statistical Interpretation of term specificity and its application in retrieval", *Journal of Documentation.,* vol. 28, pp. 11-21, 1972.

[11] B. Martins, and M. Silva, "Web Information Retrieval with Result Set Clustering", in *Proc. of Natural Language and Text Retrieval Workshop*, 2003.

[12] D. Zhang, and Y. Dong, "Semantic, Hierarchical, Online Clustering of Web Search Results", in *Proc. of the 6ᵗʰ Asia Pacific Web Conference*, China, 2001.

[13] O. Zamir, and O. Etzioni, "Web Document Clustering: A Feasibility Demonstration", in Proc. of the 19ᵗʰ Annual International SIGIR, 1998, pp. 46-54.

[14] Z. Jiang, A. Joshi, R. Krishnapuram, and Y. Yi,

[15] C. Carpineto, and G. Romano, "Exploiting the Potential of Concept Lattices for Information Retrieval with CREDO", in *Journal of the Universal Computer Science*, 2004.

[16] P. Hannappel, R. Klapsing, and G. Neumann, "MSEEC - a multi search engine with multiple clustering", in *Proc. of the 99 Information Resources Management Association International Conference*, Hershey, Pennsylvania, May, 1999.

[17] D. Lawrie, and B. Croft, "Generating hierarchical summaries for web searches", in *Proc. of 26ᵗʰ Annual International SIGIR Conf.*, Toronto, Canada, 2003.

[18] R. Kummamuru, R. Lotlikar, S. Roy, K. Singal, K. Krishnapuram, "A hierarchical monothetic document clustering algorithm for summarization and browsing search results", in *Proc. of the International Conference on WWW*, NY, USA, 2004, pp. 658–665.

[19] M. Hearts, and J. Pedersen, "Re-examining the Cluster Hypothesis: Scatter/Gather on Retrieval Reuslts", in *Proc. of the 19ᵗʰ Annual International SIGIR Conf.*, Zurich, Switzerland, 1996, pp. 76-84,

[20] M. Sanderson, and B. Croft, "Deriving concepts hierarchies from text", *in Proc. of 21ᵗʰ Annual International SIGIR Conf.* 1999, pp. 206-213.

[21] C. Zhai, X. Tong, N. Millic-Frayling, and D. Evans, "Evaluations of syntactic phrase indexing – CLARIT", NLP track report, in D.K. Harman (ed.), The fifth text retrieval conference (TREC-5). NIST Special Publication, 1997.

[22] G. Mishne, and M. Rijke, "Boosting Web Retrieval through Query Operations", In *Proc. of 27ᵗʰ ECIR*, 2005.

[23] J. Fagan,"Experiments in automatic phrase indexing for document retrieval: a comparison of syntactic and non-syntactic methods", *Ph.D. Thesis*, University of Cambridge, 1978.

[24] J. Pickens, and B. Croft, "An Exploratory Analysis of Phrases in Text Retrieval", In *Proc. of RIAO*, 2000.

[25] M. Mitra, C. Buckley, A. Singhal, and C. Cardie, "An Analysis of Statistical and Syntactic Phrases", in *Proc. of RIAO,* Canada, 1997, pp. 200-214.

[26] O. Vechtomova, and M. Karamufluoglu, "Approaches to High Accuracy Retrieval: Phrase-Based Search Experiments in the HARD Track", in *Proc. of the 13ᵗʰ Text Retrieval Conf.* Gaithersburg, 2004, pp. 16-19.

[27] R. Jones, O. Vechtomova, and G. Dias,

"Methodologies and Evaluation of Lexical Cohesion Techniques", in *Real-world Applications - Beyond Bag of Word Workshops*. 2005.

[28] G. Dias, S. Guilloré, and G. Lopes, "Language Independent Automatic Acquisition of Rigid Multiword Units from Unrestricted Text Corpora", in *Proc. of the 6th Conf.Annuelle du Traitement Automatique des Langues Naturelles*, Cargèse, France, July 12-17, 1999.

[29] J. Silva, G. Dias, S. Guilloré, and G. Lopes, "Using LocalMaxs Algorithm for the Extraction of Contiguous and Non-contiguous Multiword Lexical Units", in Proc. of the 9th Portuguese Conference in Artificial Intelligence, Évora, Portugal, September 21-24, 1999, pp. 113-132.

[30] B. Daille, "Study and Implementation of Combined Techniques for Automatic Extraction of Terminology", in *the Balancing Act Combining Symbolic and statistical Approaches to Language*, Cambridge, MIT Press, 1995.

[31] J. Justeson, "Technical Terminology: Some Linguistic Properties and an Algorithm for Identification in Text", in *IBM Research Report RC 18906 (82591)*, May, 1993.

[32] R. Kosala, and H. Blockeel, "Web Mining Research: a Survey", in *ACM SIGKDD Exploration*, Vol. 2(1), pp. 1-15, 2000.

[33] J. Rocchio, "Relevance Feedback in Information Retrieval: The Smart System – Experiments in Automatic Document Processing", Prentice-Hall., 337-354, 1971.

[34] P. Turney, "Learning Algorithms for Keyphrase Extraction", in *Information Retrieval*, Vol. 2(4), 303-336, 2000.

[35] C. Muller, X. Polanco, J. Royauté, and Y. Toussaint, " Acquisition et Structuration des Connaissances en Corpus: Eléments Méthodologiques", *Technical Report*, Institut National de Recherche en Informatique et en Automatique, 1997.

[36] A. Leouski, and B. Croft, "An evaluation of techniques for clustering search results", *Technical Report IR-76*, University of Massachusetts, 1996.

[37] M. Steinbach, "A comparison of Document Clustering techniques", in *Proc. of KDD*, University of Minnesota, 2000.

[38] P. Willet, "Recent trends in hierarchical document clustering: a critical review", *Information Processing and Management*, Vol. 24, pp. 577-97, 1988.

[39] D. Bruza, S. Dennis, R. McArthur, "Interactive internet search: keyword directory and query reformulation mechanisms compared", in Proc. of the 23th International ACM SIGIR, 2000.

Ricardo Campos has a Master Degree in Computer Science. He is a Professor at the Polytechnic Institute of Tomar where he teaches subjects related to ICT. He is a researcher at the Centre of Human Language Technology and Bioinformatics (HULTIG) and has several papers published in the fields of Web IR and e-Gov, and has been guest to participate in some research projects, conference organizations, journal and book reviews. He has been member of the program committee of several international conferences. For comprehensive access to his works, please refer to his website at http://hultig.di.ubi.pt/~ricardo

Gaël Dias holds a PhD in Computer Science in the specific area of NLP. He is an Assistant Professor of the Department of Computer Science of the University of Beira Interior (Covilhã, Portugal) where he teaches subjects related to AI and System Analysis. He is the responsible for HULTIG, researcher at the Centre of Mathematics of UBI and Invited Researcher at the L²F Group of INESC-ID in Lisbon (Portugal). He has the Erdős Number 5 classification (Erdős number project). For comprehensive access to his works, please refer to his website at http://di.ubi.pt/~ddg

Célia Nunes has a PhD in Maths. She is currently a Professor at the Maths Department of the University of Beira Interior (Covilhã, Portugal). She is a researcher at the Centre of Mathematics of the UBI, HULTIG and has several papers published in the field of statistics. She is a reviewer of Mathematical Reviewers, member of the program committee of several international conferences and has been guest to participate in some research projects and conference organizations. For comprehensive access to her works, please refer to her website at http://mat.ubi.pt/~celia

Bono Nonchev got a Degree in Applied Maths in 2008. He has gain two bronze medals at the Olympiads of Informatics and a National Diploma from excellent marks. In 2007 he worked, under a scholarship, at HULTIG. He has papers published in the field of distributed computation.