MetaCluster.PT: A Meta-Search engine for the Portuguese Web
(Extended Abstract)

Nuno Miguel Salvado Amador

1) Instituto Superior Técnico

Abstract. It is known that a single search engine only indexes a small part of the whole Web. When we search using one of these engines, we can see that some return results that others do not. Thus, combining some of the engines, lets us gain access to more indexed documents. That is the idea behind Meta-Searching. A Meta-Searcher is a search engine that searches other individual search engines and then combines the results and shows them to the user, avoiding the user to do the individual searches himself. Furthermore, the results can be grouped and presented by topics, in a process called clustering. This Thesis describes a prototype of a Meta-Search engine with clustering capabilities for the Portuguese Web.

Key words: Information Retrieval, Meta-Search, Web Clustering, K-Means

1 Introduction

Conventional search engines present many problems. One of these problems is that a single search engine cannot index the whole Web by himself. The indexed and returned results by one engine may not exist in another, for the same search terms. To help minimize this problem, a new type of search engines was invented. A Meta-Searcher is a type of search engine which queries several conventional search engines, collecting from them much more results than is possible to retrieve from a single engine [17]. Meanwhile, Web search sometimes gives us results that, despite sharing terms or words with the user query, are not semantically related with the information that the user wants. For example, a search for “jaguar” normally returns documents about cars of this brand, and about big cats. A solution for this problem is to organize the results, according to the different topics encountered in them. For the given example, we would like to separate the results in two distinct groups, one for cars of brand “jaguar”, and another for the big cat. This can be done with a process called clustering [18].

The prototype developed and presented with this Thesis, consist in a Meta-Search engine for the Portuguese Web, with result clustering capabilities.

The following sections give us an overview of the Thesis, describing: some generic and web clustering algorithms, and some pioneer systems where the algorithms were first applied; the architectures of conventional and meta-search
engines; a meta-search engine prototype for the Portuguese Web as a proposed solution; the evaluation and the experimental results obtained with the prototype; and finally, the conclusions.

2 Clustering Algorithms

Clustering is a method of unsupervised learning, i.e., without human intervention and without previous knowledge of the input data [3]. The method consists in automatically grouping elements of a data set in several groups, or clusters, with each cluster formed by equal or related elements, and in which elements of different clusters are unequal or unrelated. Web clustering [3] thus corresponds to clustering applied to a set of Web documents, in this case returned by one or several search engines. In the context of the present work, a element is a Web document or page and, a cluster is a sub-set of grouped documents, described by a topic. Web clustering helps the user understand the natural and semantic grouping of Web documents, which are grouped in several topics. It is, therefore, interesting to group Web information automatically, and it is here that the clustering algorithms reveal its importance. A good clustering algorithm tries to minimize inter-cluster similarity (between elements of different clusters) and, at the same time, maximize intra-cluster similarity (between elements of the same cluster). To determine to which cluster an element belongs, it uses a similarity function [1], which measures how an element or cluster is related or similar to another.

2.1 Hierarchical clustering

The data clustering algorithms can be hierarchical or flat. Hierarchical algorithms find successive clusters using previously established clusters. Also, we can define hierarchical relations between various clusters. For example, if we have two groups with topics “sports” and “tennis”, and next we can define that the sub-group “tennis” belongs to the more comprehensive group “sports”. Hierarchical methods can also be classified in agglomerative (also known as “bottom-up”) or divisive (“top-down”) [2]. In the agglomerative case, we start with the individual elements and successively merge each one to another, and forming the clusters. The divisive approach does the opposite: we start with the whole set, viewed as a super-cluster, and then we recursively split the clusters. In both cases the process stops when a certain criteria is met, or when we cannot further merge (agglomerative case) or split (divisive) anymore. At each iteration, the next elements to merge or to split are determined by some proper similarity (or distance) function.

2.2 Non-Hierarchical or Flat clustering

In the non-hierarchical or flat clustering, all elements and clusters are at the same level. The clusters are formed in “flat area” and, in most cases, in a partitive way.
The most cited and common example of a partitive algorithm is the *K-Means* algorithm [8]. This algorithm works as follows: initially $K$ clusters are given, and assigned $K$ centroids to represent the center of these clusters. In theory, the centroids represent the “mass center” of the cluster. Next, each element is assigned to the cluster who has his centroid closer to the element. The centroids are now re-calculated, considering this new assigned element, and the process repeats itself until no more changes are observed in the location of the centroids. The final clusters are non-hierarchical and non-overlapping. The *K-Means* linear algorithm is fast, but we must give an initial number $K$ of desired clusters, which is initially unknown in the input data set. This is a very generic algorithm, and can be applied in a number of situations. The quest for a clustering algorithm specially designed to cluster Web documents has just began.

### 2.3 Dealing with text: Web clustering Algorithms

In order to cluster text, some noise must be removed first from the input documents. Invalid characters, HTML tags, symbols, some numbers... all of these are considered noise and must be filtered and removed from the input text.

Next, some pre-processing must also be performed before applying the real clustering algorithms. This pre-processing consists of two important phases, commonly used in the areas of text mining and Information Retrieval: stopword marking (or removal), and stemming [1]. Stopwords are frequent words present in the text, that do not add any valuable information, and thus can be discarded, or marked to be ignored. For example, “de” is a typical Portuguese stopword and, if not removed, we might end up with some useless clusters around this word after clustering. Another important phase is stemming, which is the process for reducing inflected (or sometimes derived) words to their stem, base or root form - generally a written word form. This allows algorithms to deal with the various forms of a word, in a simple way, treating only the stem and not all the word variants. After pre-processing, the actual clustering algorithms can now be applied.

The *TRC* (Tolerance Rough Clustering) algorithm [7], is based on the *K-Means* algorithm. However, instead of assigning each document to a specific cluster, the algorithm tries to find tolerance classes built with the similarities found between documents. This Soft-Clustering contrasts with the Hard-Clustering provided by *K-Means*, because a document can now belong to more than one class (cluster). The descriptions for the clusters are formed using word $n$-grams (phrases formed with up to $n$ words), which are collected from documents in each class.

A pioneer algorithm, called *Scatter-Gather* [5, 16], was one of the first specially designed to cluster Web documents. *Scatter-Gather* is similar to the *K-Means*: it is also necessary to known in advance the number $K$ of desired clusters but, unlike *K-Means*, the initial centroids are not randomly chosen. This is done with the help of two routines, *Fractionation* and *Buckshot*, where both find $K$ centroids on the document set. The centroids found are then used as input initial centroids for the *K-Means* algorithm.
One of the problems with both \textit{K-Means} and \textit{Scatter-Gather} algorithms, is the search of a good method to choose the labels to describe a cluster topic. This description must be a phrase carefully chosen from the input documents, but at the same time should allow to describe accurately the documents contained in the cluster.

In an attempt to solve this problem, the \textit{STC} (Suffix Tree Clustering) \cite{19} does not treat the documents just like a bag of words, like the previous algorithms did. Instead, it treats the documents like a set of phrases, making use of the word proximity and order information present in the text. To achieve this, the \textit{STC} uses an suffix tree to efficiently identify base clusters which share common phrases. These base clusters receive a score based on the number of documents and words they represent. Only base clusters with a score greater than a pre-configured threshold are promoted to real clusters. Finally, these real clusters are merged to final clusters. If the similarity between them is greater than another pre-configured threshold, which at the end prevents having very similar or duplicate clusters. Besides retaining and using word position, the \textit{STC} is a linear algorithm and very efficient, which allows to cluster documents as soon as they are returned from the search engines.

Another algorithm which tries to solve the cluster description problem is the \textit{Lingo} algorithm \cite{11}. However, this algorithm takes another approach, called “Description comes first”. The biggest motivation for \textit{Lingo}, is that classical approaches only describe clusters after these have been formed, resulting sometimes in poor topic perceptibility to users. Thus, the \textit{Lingo} algorithm first looks for descriptions directly in the snippets of the input results, in an attempt to form good cluster descriptions, and only then individual documents are added to each cluster, this is, the clustering phase itself.

3 Search engine Architectures

There are two common types of search engines. The \textit{conventional search engine} (SE), and the \textit{Meta-Search engine} (MSE).

A conventional search engine is basically an Information Retrieval system \cite{4}, whose main components are:

- \textit{Web Crawler} (also known as \textit{Web Spider} or \textit{Web Robot}) – an independent program which searches the Web for documents, and stores them in the SE \textit{Repository}.
- \textit{Crawl Control} – a component which determines the \textit{Web Crawler} behavior.
- \textit{Repository} – database which stores the documents and pages before indexation and classification.
- \textit{Indexer} and \textit{Collection Analyzer} – periodically, these components are responsible for the classification and indexation of the documents, storing the results in separated databases. These databases are the core of the conventional SE.
- \textit{Query Engine} – receives and processes the user typed queries terms.
– **Ranking** – sorts and processes the result lists, before presenting them to the user.

In these engines, the coverage of the *Web Crawler* is limited. It is impossible for a single SE to index the whole Web. This is where the Meta-Search engine can help.

An MSE is a search tool that sends user requests to several other search engines and aggregates the returned results, processing and/or sorting them in some manner, and finally presenting them to the user. Physically, the system acts as a mediator between the user and the various search engines. The MSE architecture is composed of various components [9]:

– **User Interface** – processes user input queries and presents the results.
– **Database Selector** – selects the best search engines to query, in order to best fulfill the user request.
– **Document Selector** – like the previous components, this one selects the best documents from the returned set, to best fulfill the user request.
– **Query Dispatcher** – this component receives the user query terms, and translates and prepares them, before sending them to each of the conventional SEs.
– **Result Merger** – merges and sorts by some criteria the returned results before presenting them to the user.

By combining the results returned by several search engines, we are increasing our coverage of the Web, and also the chance to find something relevant for our search. After collecting the results, the MSE can balance the weight of each search engine and each document, before processing and sorting the final result list presented to the user.

### 4 Search engines with clustering capabilities

The first implementation of the *Scatter-Gather* algorithm, previously described, was in the system with the same name [5, 16]. The *Scatter/Gather* system was a pioneer in text clustering. It allowed browsing the news published in the *New York Times News Service*. The system knew from the outset the entire set of documents, in which he discovers the dominant topics. The user did not have to provide terms to start the search, which is initiated by the system, and this is considered one of its major advantages [16, 15]. Next, the dominant topics are presented to the user, which in turn selects one or more he finds relevant. At each iteration, the search is refined for the chosen topics, and the cycle continues until the search returns the individual documents. Later, Hearst and Pedersen [6] applied the *Scatter-Gather* algorithm for clustering the Web results returned by a search engine.

Another system, the *Grouper* [20, 21] system, was one of the first specially designed for Web search, with Meta-Search and clustering capabilities. The clustering algorithm implemented in the *Grouper* was the *STC*, which clustered the
list of results provided by another system, the HuskySearch. The results provided by HuskySearch were previously indexed by a Meta-Search engine, called MetaCrawler, which in turn performs parallel requests to various search engines. These three systems were also used for other studies, mainly in the areas of Information Retrieval and Artificial Intelligence.

The first implementation of the Lingo algorithm was on the Carrot framework, developed by David Weiss et al. [11, 14, 13, 12], and whose motivations were inspired on the previously work done in the Grouper system. The Carrot system is an academic Java Open Source project, which facilitates the implementation, study and development of new search and processing methods, and new clustering algorithms for dealing with Web pages. In version 2.x, it possesses a modular architecture, which is essentially composed by four types of components:

- **Input** – the major role of this type of component is to generate or collect data, which may be based on user input, for further processing by the other components.
- **Filter** – these components processes and changes the input data in a specific way. Examples of this type of component include: stopword marking and removal, stemming and clustering algorithms, property extraction, etc.
- **Output** – at the end of the processing chain, these components consume the information of the previous stages. A typical use of this type of component is to prepare data to present to the user.
- **Controller** – a special type of component that coordinates the other types, forming a processing chain.

Later, in version 3.0 of Carrot, the concept of “component attribute” was introduced. This attribute is a special variable which affects the behavior of the component in a specific way, or carries the output results of the component. The attributes of the components in the processing chain are managed by the controller component, which also manages the life cycle of a component in the context of a request.

At present, the Clusty\(^1\) Meta-Search engine represents the state-of-the-art when it comes to search engines with Web clustering and Meta-Search capabilities. The Clusty system is a direct descendant of the Vivisimo system, many times awarded with “best Meta-Search award” between 2001 and 2003, by the online magazine SearchEngineWatch\(^2\). Their success was due in part to the quality of the topics and its formed clusters, and satisfaction of their users.

## 5 Proposed solution

After analyzing the case of the Portuguese Web, it was found that it is still behind with regard to search engines, and it is more common to find portals with

---

\(^1\) http://clusty.com

\(^2\) http://www.searchenginewatch.com
“directories”, which only offer lists of topics ranked manually. The Portuguese search engines are few, and Meta-Search engines practically do not exist. Under these assumptions, the proposed solution is to change the Carrot framework enabling it to function in the context of the Portuguese web. Therefore, a Portuguese Meta-Searcher with clustering capabilities was created, and was properly parameterized to deal with the particularities of the Portuguese language. The Carrot version 3.0 framework was used in prototype construction, and the principal changes include:

- **Input components** – new document sources for Portuguese search engines were implemented, and the existing ones were modified to provide Portuguese (and Brazilian) results. In the search engines with no search API, the results are collected directly by parsing the HTML pages returned. For implementing the “Web” Meta-Search document source, 4 conventional search engines are used: Sapo, Google, NetIndex and Tumba.

- **Pre-processing** – the pre-processing was modified, in order to use a list of Portuguese (and Brazilian) stopwords. The majority of words present in this list, were collected from the CHAVE collection, which is maintained by Linguateca Portuguesa. Also, stemming algorithms for dealing with the Portuguese language, including Porter PT and RSLP [10], were included in the prototype.

- **Processing** – in this stage, the clustering algorithms STC and Lingo, already present in the Carrot version 3.0 were used. In the course of this Thesis, the TRC algorithm was also migrated from the version 2.x of Carrot, and adapted to the prototype.

- **Output and Visualization** – the Carrot interface was entirely translated to Portuguese, and new options were implemented. A specially modified interface was also designed and used, for collecting the results submitted by a user survey, which were then used for prototype evaluation. When presenting the output, the results list returned by the “Web” Meta-Search source had to be merged and sorted, and a simple algorithm to remove duplicates was implemented.

Next, the prototype evaluation and experimental results are described.

6 Evaluation and experimental results

A modified interface was used for the experimental evaluation. The various options and algorithm parameters were hidden, under the form of 20 different types of search buttons, in order to not influence the users responses to the survey. This survey, consisted essentially on two links, one for result evaluation, and the other for cluster evaluation. In the result evaluation, at most the first 20 results are presented to the user, who then marks each result as relevant or not.

In the cluster evaluation, all topics are presented to the user, who then marks each topic as relevant or not and, additionally, indicates if the topic description is self-explanatory, thus allowing to infer the contents of each cluster.

Next, the typical precision formula of Information Retrieval is used, in order to evaluate the precision of the top 20 results, and the precision of different tested values for some parameters of the clustering and stemming algorithms.

Results show that, the precision of the results returned by the prototype stays above 60% for the first 10 results, which are the ones typically viewed by users, as shown in Figure 1.

Fig. 1. Average Precision for the Top 5, 10 and 20 results.

In the clustering algorithms, both the \textit{STC} and \textit{Lingo} precisions and topic descriptions are found to be superior to the ones achieved by \textit{TRC}. Also, different tested values for some of the clustering algorithms parameters, confirms that the same values used for the English language, in some cases, may not achieve such as good precision when used with the Portuguese language. Some parameters do not influence the achieved precision at all. Other parameters, like the \textit{Merging Threshold} from both \textit{Lingo} and \textit{STC}, have a direct impact in topic formation and precision.

In case of the stemming algorithms, results show that, when not using stemming at all, the precision is higher for the first 10 topics, and is advised to use stemming algorithms only when dealing with more than 10 topics. In terms of good descriptions for the clusters, both \textit{Snowball PT} and \textit{Porter PT} stemming algorithms achieved a precision superior to the one of \textit{RSLP}.

In terms of efficiency, \textit{Lingo} and \textit{STC} are very efficient, with a linear response in time with the increase of the input documents. The \textit{TRC}, on the other hand, revealed a quadratic response and should not be used to cluster much more than 100 or 150 input documents, in order to keep a low response time to the user.

The set of documents returned by the “Web” Meta-Search source was also evaluated. In this test, maintaining the same number of 100 requested results and varying only the query terms with 10 different searches, the effectiveness of
the Meta-Search always stayed above 83%. This number was obtained with a normalized function, used to measure how distinct the set of documents of the various sources are. In this function, the number of 100% means that all results are distinct, and this constitutes the ideal scenario for the Meta-Search.

7 Conclusions

The continuous and exponential growth of the available information on the Web poses new and challenging problems to search engines. It is a fact that it is impossible for a single search engine index the whole Web. A solution to this problem is to use a particular type of search engine, called Meta-Search engine, which is a system that act as a mediator between the user search requests and various conventional search engines. Therefore, the user only has to deal with one interface, and the system is responsible for translate the user terms to query each individual engine, collect all results, and finally merge and sort them, before showing them to the user. With the help of Meta-Search, and the combined indexed results of the various search engines, we are augmenting the Web coverage, and accessing more information. In this Thesis, the architectures of both conventional search engine and Meta-Search engine are described, as well as its main components.

On the other hand, the search engines return too many results, and the user generally only looks for the first ones, and thus discarding relevant results to his search. Therefore, it will be useful if we could group the results by semantic topics. This can be done by a clustering algorithm, which is a method of unsupervised learning and, without human intervention, can group similar documents in clusters. This Thesis offers a theoretical introduction to clustering algorithms, focusing in Web clustering, and also describes some of the pioneer systems where these algorithms were first implemented.

When studying the state of the art, we found that the Portuguese Web is still very behind, regarding Meta-Search engines with clustering capability. Based on these assumptions, this thesis proposes a prototype with this functionalities. The proposed system is entirely based and developed on Open Source components available today on the Web, starting with the help of the Carrot framework. The components were modified in order to only gather and process results in Portuguese.

After development, the prototype was tested with real users, who responded to a survey, and voted for relevant results and topics obtained through their own searches. Tests show that, the algorithm parameter values for the English language may not be the best for the Portuguese language. Also, precision and efficiency tests to stemming and clustering algorithms were provided. The document set results returned by Meta-Search is also evaluated, and is shown to be effective.
References


