STC+ and NM-STC: Two Novel Online Results Clustering Methods for Web Searching

Stella Kopidaki, Panagiotis Papadakos, and Yannis Tzitzikas

Institute of Computer Science, FORTH-ICS, GREECE,
Computer Science Department, University of Crete, GREECE
{skopidak, papadako, tzitzik}@ics.forth.gr

Abstract. Results clustering in Web Searching is useful for providing users with overviews of the results and thus allowing them to restrict their focus to the desired parts. However, the task of deriving single-word or multiple-word names for the clusters (usually referred as *cluster labeling*) is difficult, because they have to be syntactically correct and predictive. Moreover efficiency is an important requirement since results clustering is an online task. *Suffix Tree Clustering (STC)* is a clustering technique where search results (mainly snippets) can be clustered fast (in linear time), incrementally, and each cluster is labeled with a phrase. In this paper we introduce: (a) a variation of the STC, called STC+, with a scoring formula that favors phrases that occur in document titles and differs in the way base clusters are merged, and (b) a novel algorithm called NM-STC that results in hierarchically organized clusters. The comparative user evaluation showed that both STC+ and NM-STC are significantly more preferred than STC, and that NM-STC is about two times faster than STC and STC+.

1 Introduction

Web Search Engines (WSEs) typically return a ranked list of documents that are relevant to the query submitted by the user. For each document, its title, URL and *snippet* (fragment of the text that contains keywords of the query) are usually presented. It is observed that most users are impatient and look only at the first results. Consequently, when either the documents with the intended (by the user) meaning of the query words are not in the first pages, or there are a few dotted in various ranks (and probably different result pages), it is difficult for the user to find the information he really wants. The problem becomes harder if the user cannot guess additional words for restricting his query, or the additional words he chooses are not the right ones for restricting the result set.

A solution to these problems is *results clustering* which provides a quick overview of the search results. It aims at grouping the results into topics, called *clusters*, with predictive names (labels), aiding the user to locate quickly one or more documents that otherwise he wouldn't practically find especially if they are low ranked (and thus not in first result pages). Results clustering algorithms should satisfy several requirements. First of all, the generated clusters should

be characterized from high intra-cluster similarity. Moreover, results clustering algorithms should be efficient and scalable since clustering is an online task and the size of the answer set can vary. Usually, only the $top - L$ documents are clustered in order to increase performance. In addition, the presentation of each cluster should be concise and accurate, allowing users to detect what they need quickly. *Cluster labeling* is the task of deriving readable and meaningful, single-word or multiple-word names for clusters, in order to help the user to recognize the clusters/topics he is interested in. Such labels must be predictive, allowing users to guess the contents of each cluster, descriptive, concise and syntactically correct. Finally, it should be possible to provide high quality clusters based on small snippets rather than the whole documents.

Clustering can be applied either to the original documents (like in [3, 10, 7]), or to their (query-dependent) snippets (as in [25, 23, 17, 6, 27, 8, 20]). For instance, clustering meta-search engines (MWSEs) (e.g. clusty.com) use the results of one or more search engines (e.g. Google, Yahoo!), in order to increase coverage/relevance. Therefore, meta-search engines have direct access only to the snippets returned by the queried search engines. Clustering the snippets rather than the whole documents makes clustering algorithms faster. Some clustering algorithms [6, 4, 24] use internal or external sources of knowledge like Web directories (e.g. DMoz[1], Yahoo! Directory), dictionaries (e.g. WordNet) and thesauri, online encyclopedias (e.g. Wikipedia[2]) and other online knowledge bases. These external sources are exploited to identify key phrases that represent the contents of the retrieved documents or to enrich the extracted words/phrases in order to optimize the clustering and improve the quality of cluster labels.

Suffix Tree Clustering (STC) [25] is a clustering technique where search results (mainly snippets) are clustered fast (in linear time), incrementally, and each cluster is labeled with a common phrase. Another advantage of STC is that it allows clusters to overlap. In this work we introduce: (a) a variation of the STC, called STC+, with a scoring formula that favors phrases that occur in document titles and differs in the way base clusters are merged, and (b) a novel algorithm called NM-STC (No Merge STC) that adopts a different scoring formula, it does not merge clusters and results in hierarchically organized labels. The advantages of NM-STC are: (a) the user never gets unexpected results, as opposed to the existing STC-based algorithms which adopt overlap-based cluster merging, (b) it is more configurable w.r.t. desired cluster label lengths (STC favors specific lengths), (c) it derives hierarchically organized labels, and (d) it favors occurrences in titles (as STC+) and takes into account IDFs, if available. The empirical evaluation showed that users prefer the STC+ and NM-STC than the original STC. NM-STC is currently in use by `Mitos` WSE [13][3].

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 formulates the problem and introduces notations. Section 4 describes

[1] www.dmoz.org
[2] www.wikipedia.org
[3] http://groogle.csd.uoc.gr:8080/mitos/

the clustering algorithms and Section 5 reports experimental results. Finally Section 6 concludes and identifies issues that are worth further research.

2 Related Work

At first we discuss clustering approaches based on document vectors and then approaches based on snippets (focusing on STC). Finally, we discuss cluster presentation and user interaction.

Document Vector-based Approaches Traditional clustering algorithms either flat (like K-means) or hierarchical (agglomerative or divisive) are not based on snippets but on the original document vectors and on a similarity measure. For instance, a relatively recent approach is Frequent Itemset Hierarchical Clustering (FIHC) [7] which exploits the notion of frequent itemsets used in data mining. In brief, such approaches can be applied only on a stand alone engine (since they require accessing the entire vectors of the documents) and they are computationally expensive.

Snippet-based Approaches Snippet-based approaches rely on snippets and there are already a few engines that provide such clustering services (Clusty[4] is probably the most famous one). Suffix Tree Clustering (STC) [25] is a key algorithm in this domain and is used by Grouper [26] and *Carrot*^{2} [23, 17] MWSEs. It treats each snippet as an ordered sequence of words, it identifies the phrases (ordered sequences of one or more words) that are common to groups of documents by building a suffix tree structure, and it returns a flat set of clusters that are naturally overlapping. Several variations of STC have been proposed. For instance, the trie can be constructed with the N-grams instead of the original suffixes. The resulting trie has lower memory requirements (since suffixes are no longer than N words) and its building time is reduced, but less common phrases are discovered and this may hurt the quality of the final clusters. Specifically, when N is smaller than the length of true common phrases the cluster labels can be unreadable. To overcome this shortcoming [11] proposed a join operation. A variant of STC with N-gram is STC with X-gram [20] where X is an adaptive variable. It has lower memory requirements and is faster than both STC with N-gram and the original STC since it maintains fewer words. It is claimed that it generates more readable labels than STC with N-gram as it inserts in the suffix tree more true common phrases and joins partial phrases to construct true common phrases, but no user study results have been reported in the literature. The performance improvements reported are small and from our experiments the most time consuming task is the generation of the snippets (not the construction of the suffix tree). Another approach based on STC is ESTC (Extended STC) [2], an extension of STC appropriate for application over the full texts (not snippets). To reduce the (roughly two orders of magnitude) increased number of clusters, a different scoring function and cluster selection algorithm is adopted. The cluster selection algorithm is based on a greedy search algorithm aiming

[4] www.clusty.com

at reducing the overlap and at increasing the coverage of the final clusters. We do not share the objective of reducing overlap as in practice documents concern more than one topic. The comparison of ESTC with the original STC was done using a very small cluster set (consisting of only two queries) and no user study has been performed. Moreover, the major part of the evaluation was done assuming the entire textual contents of the pages (not snippets), or on snippets without title information. Summarizing, clustering over full text is not appropriate for a (Meta) WSE since full text may not be available or too expensive to process. Other extensions of STC for oriental languages and for cases where external resources are available are described in [28, 21].

Another snippet-based clustering approach is *TermRank* [8]. TermRank succeeds in ranking discriminative terms higher than ambiguous terms, and ambiguous terms higher than common terms. The $top - T$ terms, can then be used as feature vectors in K-means or any other Document Vector-based clustering algorithm. This approach requires knowing TF, it does not work on phrases (but on single words) and no evaluation results over snippets have been reported in the literature.

Another approach is Findex [12], a statistical algorithm that extracts candidate phrases by moving a window with a length of $1..|P|$ words across the sentences (P), and fKWIC which extracts the most frequent keyword contexts which must be phrases that contain at least one of the query words. In contrast to STC, Findex does not merge clusters on the basis of the common documents but on the similarity of the extracted phrases. However, no comparative results regarding cluster label quality have been reported in the literature.

Finally, there are snippet-based approaches that use *external resources (lexical or training data)*. For instance, SNAKET [6] (a MWSE) uses DMoz web directory for ranking the *gapped sentences* which are extracted from the snippets. *Deep Classifier* [24] trims the large hierarchy, returned by an online Web directory, into a narrow one and combines it with the results of a search engine making use of a discriminative naive Bayesian Classifier. Another (supervised) machine learning technique is the *Salient Phrases Extraction* [27]. It extracts *salient phrases* as candidate cluster names from the list of titles and snippets of the answer, and ranks them using a regression model over five different properties, learned from human training data. Another approach that uses several external resources, such as WordNet and Wikipedia, in order to identify useful terms and to organize them hierarchically is described in [4].

Cluster Presentation & User Interaction Although cluster presentation and user interaction approaches are somehow orthogonal to the clustering algorithms employed, they are crucial for providing flexible and effective access services to the end users. In most cases, clusters are presented using lists or trees. Some variations are described next. A well known interaction paradigm that involves clustering is Scatter/Gather [3, 10] which provides an interactive interface allowing the users to select clusters, then the documents of the selected clusters are clustered again, the new clusters are presented, and so on. In our case we adopt the interaction paradigm of *dynamic taxonomies* [16] as it is the

de facto standard in e-commerce (and users are already familiar with), and it can enable guided browsing over *explicit* and *mined* metadata. The automatically derived cluster labels fall into the latter category.

3 Problem Statement and Notations

We consider important the requirements of *relevance, browsable summaries, overlap, snippet-tolerance, speed* and *incrementality* as described in [25]. Regarding the problem of cluster labeling we have observed that: (a) *long labels are not very good* (e.g. not convenient for the left frame of a WSE, or for accessing the WSE through a mobile phone) (b) *very short labels (e.g. single words) are not necessarily good* (e.g. longer labels could be acceptable, or even desired, in a system that shows the cluster labels in a horizontal frame) (c) *an hierarchical organization of labels can alleviate the problem of long labels*, and (d) *the words/phrases appearing in titles are usually better (for cluster labeling) than those appearing only in snippets*. Observations (a) and (b) motivate the need for configuration parameters. Observations (c) and (d) motivate the algorithms STC+ and NM-STC that we will introduce.

Configuration Parameters We have realized that several configuration parameters are needed for facing the needs of a modern WSE. We decided to adopt the following: K : number of top elements of the answer to cluster, LL_{max} : max cluster Label Length, LL_{min} : min cluster Label Length, and NC_{max} : max Number of Clusters. Obviously it should be $NC_{max} < K$. However the size of the current answer should also be taken into account. Specifically if $ans(q)$ is the answer of the submitted query, then we shall use A to denote the first K elements of $ans(q)$. However, if $|A| < K$ then we assume that $K = |A|$.

Notations We use Obj to denote the set of all documents, hereafter objects, indexed by a WSE, and A to denote the top-K elements of the current answer as defined earlier (i.e. $A \subseteq Obj$ and $|A| = K$).

We use W to denote the set of words of the entire collection, and $W(A)$ to denote the set of the words that appear in a set of documents A (this means that W is a shortcut for $W(Obj)$).

Let $A = \{a_1, \ldots, a_K\}$. For each element a_i of A we shall use $a_i.t$ to denote the title of a_i, and $a_i.s$ to denote the snippet of a_i. Note that the elements of $W(A)$ are based on both titles and snippets of the elements of A.

If a is a text, then we shall use $P(a)$ to denote all phrases of a that are *sentence suffixes*, i.e. start from a word beginning and stop at the end of a sentence of a. For example P("this is a test") = {"this is a test", "is a test", "a test", "test"}, while P("this is. A test") = {"this is", "is", "A test", "test"}.

We shall use $P(A)$ to denote all phrases of the elements of A, i.e. $P(A) = \bigcup_{a \in A}(P(a.t) \cup P(a.s))$.

If p is a phrase we shall use $Ext(p)$ to denote the objects (of A) to which p appears, i.e. $Ext(p) = \{a \in A \mid p \in a\}$. Also, we shall use $w(p)$ to denote the set of words that phrase p contains.

4 STC, STC+ and NM-STC

Our goal is to improve STC, specifically: (a) to improve the quality of cluster labels by exploiting more the titles (document titles can give more concise labels), (b) to define a more parametric algorithm for facing the requirements of modern WSEs, and (c) to derive hierarchically organized labels. Specifically below we describe the original STC, a variation that we have devised called STC+, and a new algorithm called NM-STC.

Original STC In brief, Suffix Tree Clustering (STC) uses the titles and snippets of the search results in order to create groups of documents that share a common phrase. Specifically, titles and snippets, after a preprocessing phase, are inserted in a generalized suffix tree structure which allows us to identify the common phrases and the documents they appear. The suffix tree [22, 9] is a data structure that can be constructed in linear time with the size of the collection, and can be constructed incrementally as the documents are being read [19]. A set of documents that share a common phrase is called *base cluster*. Finally, a merging step of base clusters (based on the overlap of their documents) leads to the final clusters which are scored and presented to the user.

In more detail, the algorithm starts with the suffix tree construction. For each sentence of the input data all suffixes are generated and are inserted into the suffix tree. Each node of the tree that contains two or more documents is a base cluster. Each base cluster that corresponds to a phrase p is assigned a score which is calculated with the following formula:

$$score(p) = |\{a \in A \mid p \in a.t \text{ or } \ p \in a.s\}| * f(effLen(p))$$

where $effLen(p)$ is the effective length of label p defined as:

$$effLen(p) = |w(p)| - |common(p)| \text{ where}$$
$$common(p) = \{w_i \in p \mid df(w_i, A) \leq 3 \text{ or } \ \frac{df(w_i, A)}{|A|} > 0.4\}$$

where $df(w_i, A) = |\{d \in A \mid w_i \in d\}|$.

The function f (that takes as input the effective length), penalizes single words, is linear for phrases with effective length from two to six words, and is constant for bigger phrases, specifically:

$$f(effLen(p)) = \begin{cases} 0.5 & \text{if } effLen(p) \leq 1 \\ effLen(p) & \text{if } 2 \leq effLen(p) \leq 6 \\ 7.0 & \text{if } effLen(p) > 6 \end{cases}$$

Afterwards, the overlap is calculated for all pairs of base clusters. Overlap is defined with a binary similarity measure. The similarity between two base clusters C_i and C_j is defined as $sim(C_i, C_j, 0.5)$ where:

$$sim(C_i, C_j, thres) = \begin{cases} 1 & \text{if } \frac{|C_i \bigcap C_j|}{|C_i|} > thres \text{ and } \ \frac{|C_i \bigcap C_j|}{|C_j|} > thres \\ 0 & \text{otherwise} \end{cases}$$

The next step is the merging of the base clusters. In brief, each final cluster contains all base clusters that can be merged (two base clusters can be merged if their similarity equals 1). As a result the document set of a final cluster is the union of its base clusters' document sets and its cluster label is the label of the base cluster with the highest score. Due to cluster merges there can be documents that do not contain the label p. Let $C(p)$ be the document set of a cluster label p. The exact scoring formula for a final cluster is $score(p) = |C(p)| * f(effLen(p))$. Finally, clusters are sorted according to their score and are presented to the user.

STC+: A Variation of STC Here we describe a variation of STC which differs in the way that clusters are scored and in the way base clusters are merged. Specifically, we adopt the following scoring formula:

$$score(p) = (|\{a \in A \mid p \in a.t\}| + |\{a \in A \mid p \in a.t \text{ or } \ p \in a.s\}|) * f(effLen(p))$$

This formula favors phrases that occur in titles. In addition, we have modified the function f. Our variation penalizes single words and phrases that their effective length is bigger that 4 words, and is linear for phrases with effective length two to four words. These values are a good compromise between the reported results of the user study at Section 5, favoring small phrases, and the avoidance of single-word labels. Specifically our function f is defined as:

$$f(effLen(p)) = \begin{cases} 0.5 & \text{if } effLen(p) \leq 1 \text{ or } effLen(p) > 4 \\ effLen(p) & \text{if } 2 \leq effLen(p) \leq 4 \end{cases}$$

Regarding the computation of the similarity measure (that determines cluster merging) we consider as threshold the value 0.4 instead of 0.5. From our experience, this value creates fewer and bigger clusters and solves some problematic cases of the original STC. For example, a base cluster with 2 documents that is compared with a base cluster with 4 documents cannot be merged even if they have 2 common documents, because $2/4 = 0.5$. Therefore we used $sim(C_i, C_j, 0.4)$. A lower than 0.4 threshold would decrease the *label precision* as it will be explained in Section 5. Note that the title set of a final cluster is the union of its base clusters' title sets. Let $T(p)$ be the set of titles of a cluster label p. The exact scoring formula for a final cluster is $score(p) = (|T(p)| + |C(p)|) * f(effLen(p))$.

NM-STC: A new Clustering Algorithm Here we introduce an algorithm called NM-STC (No Merge Suffix Tree Clustering). As in STC, we begin by constructing the suffix tree of the titles and snippets. Then we score each node p of that tree. Let p be a phrase (corresponding to a node of the suffix tree). Below we define four scoring functions:

$$\begin{aligned} score_t(p) &= |\{a \in A \mid p \in a.t\}| \\ score_s(p) &= |\{a \in A \mid p \in a.s\}| \\ score_{ts}(p) &= score_t(p) * |A| + score_s(p) \\ score_{tsi}(p) &= score_t(p) * |A| * N + score_s(p) * N + PIDF(p) \end{aligned}$$

PIDF stands for Phrase IDF and N is the total number of indexed documents ($N = |Obj|$). If p is a single word (w), then $PIDF(p)$ is the IDF of w (i.e. $IDF(w) = \frac{N}{|\{d \in Obj \mid w \in d\}|}$). If p is a phrase consisting of the words $\{w_1, \ldots, w_m\}$, then PIDF is the average IDF of its words, i.e.

$$PIDF(p) = \frac{1}{m} \sum_{i=1}^{m} IDF(w_i)$$

or alternatively $PIDF(p) = \max_{w \in p}(IDF(w))$. In our experiments we used the average IDF. The IDF can be computed based on the entire collection if we are in the context of a single WSE. In our case, the index of `Mitos` stores only the stems of the words, so $IDF(w)$ is estimated over the stemmed words. If we are in the context of a MWSE, then IDF could be based on external sources, or on the current answer[5].

NM-STC uses the $score_{tsi}(\cdot)$ scoring formula. This scoring function actually quantifies a qualitative preference of the form $title \rhd snippet \rhd PIDF$, where $\rhd$ denotes the priority operator [1]. Notice that PIDF has the lowest priority. It is used just for breaking some ties. From our experiments, the number of broken ties is low, so it does not affect significantly the results. Also, $score_{tsi}(\cdot)$ can be applied on STC+ instead of its scoring formula.

NM-STC at first scores all labels of the suffix tree using the function $score_{tsi}(\cdot)$. Subsequently it selects and returns the top-NC_{max} scored phrases. Let B be the set of top-NC_{max} scored phrases. Note that it is possible for B to contain phrases that point to the same objects, meaning that the extensions of the labels in B could have big overlaps. In such cases we will have low "coverage" of the resulting clustering (i.e. the set $\cup_{p \in B} Ext(p)$ could be much smaller than A).

Recall that STC merges base clusters having a substantial overlap in order to tackle this problem. However that approach leads to labels whose extension may contain documents that do not contain the cluster label (in this way users get unexpected results). Instead NM-STC follows a different approach that is described in the sequel, after first introducing an auxiliary notation. If $n(p)$ and $n(p')$ denote the nodes in the suffix tree that correspond to phrases p and p' respectively, we shall say that p is narrower than p', and we will write $p < p'$, iff $n(p)$ is a descendent of $n(p')$, which means that p' is a prefix of p. For instance, in our running example of Figure 1 we have $n(\text{"}a\ b\text{"}) < n(\text{"}a\text{"})$.

Returning to the issue at hand, our approach is the following: We fetch the top-NC_{max} labels and we compute the *maximal* elements of this set according to $<$. In this way we get the more broad labels (among those that are highly scored). If their number is less than NC_{max} then we fetch more labels until reaching to a set of labels whose maximal set has cardinality NC_{max}. So the algorithm returns the smaller set of top-scored phrases B that satisfies the equation $|maximal_<(B)| = NC_{max}$ if this is possible (even if B is the set of all nodes of the suffix tree, it may be $|maximal_<(B)| < NC_{max}$).

[5] $IDF(w) = \frac{|A|}{|\{d \in A \mid w \in d\}|}$

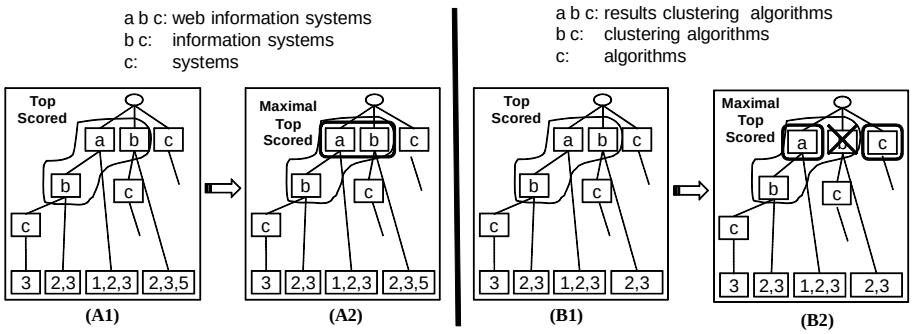

Fig. 1. Two examples of NM-STC

The extra labels fetched (i.e. those in $B \setminus maximal_{<}(B)$) are exploited by the GUI for providing an *hierarchical organization* of the labels (where the user can expand the desired nodes to see their immediate children and so on). Consider the example in Figure 1.(A1), and assume that $NC_{max} = 2$. The set of top-3 scored labels whose maximal elements are two are marked (as shown in Figure 1.(A2)). At the GUI level, the user can expand a and see the label b.

The algorithm is sketched bellow. It takes as input a tree (the suffix tree) and returns another tree (the cluster label tree). Of course it also takes as input the configuration parameters, as well as the current query q.

Alg. $NM - STC$
Input: sf:SuffixTree, NC_{max}, LL_{min}, LL_{max}, q
Output: cluster label tree
(1) ScoreLabelsOf(sf)
(2) ZeroScoreLabelsEqualTo(sf,q)
(3) ZeroScoreLabelsLabelSize(sf,LL_{min}, LL_{max})
(4) topLabs = getTopScored(sf, NC_{max})
(5) Done=False
(6) while Done=False
(7) maxTopLabs = maximal$_{<}$(topLabs)
(8) maxTopLabls = ElimSubPhrasesSameExt(maxTopLabs)
(9) missing = NC_{max} - |maxTopLabs|
(10) if (missing>0)
(11) topLabs = topLabs ∪ getNextTopScored(sf,missing)
(12) else Done=True
(13) end while
(14) return topLabs, $<_{|topLabs}$

If a cluster label p contains only query words (i.e. $w(p) = w(q)$), then we exclude it from consideration, as such labels would be useless for the users. This is done by zeroing the scores of such labels (step (2)). At step (3) we zero the scores of the labels that do not satisfy the LL_{min} and LL_{max} constraints. The function getTopScored(sf, NC_{max}) returns the NC_{max} most highly scored nodes. At step (8) we remove from the list of maximal labels those that are subphrases of other labels and contain the same documents. Specifically, if $w(p) \subseteq w(p')$ and $Ext(p) = Ext(p')$ then we exclude p. This is shown in the example illustrated in

Figure 1.(B1 and B2): the node b is discarded because it has the same extension with the node b that is child of a. The function getNextTopScored(sf, M) returns the next M labels in the ranked list of labels (that are not already consumed).

5 Experimental Evaluation

Implementation The algorithms have been implemented over `Mitos` [15, 14, 13] [6]. The snippets in our experiments were quite small: up to two sentences, each one consisting of 11 words maximum. The results of clustering are presented to the user using the `FleXplorer` API [18], that supports the interaction paradigm of *dynamic taxonomies*. The hierarchy of cluster labels (by NM-STC) can be considered as a subsumption relation since it satisfies $p < p' \implies Ext(p) \subseteq Ext(p')$, i.e. if p is child of p' then the objects associated with p are subset of those associated with p', and this allows exploiting the interaction paradigm of dynamic taxonomies. At the presentation layer the user initially views the maximal elements of the cluster label tree along with the number of $|Ext(p)|$ and a symbol indicating whether that node has children. By clicking on one of these nodes the direct children of that node appears too. The process of unfolding (expanding) labels resembles the process of extending a natural language phrase. By construction all these phrases are syntactically correct.

Evaluation by Users We conducted a comparative evaluation over `Mitos`. We defined 16 queries of different sizes consisting of small (single words), medium (2 to 3 words), and big (4 or more words) queries[7]. The queries were randomly chosen and their results sizes range from 14 to 5029 hits. The queries were given to 11 persons (from 22 to 30 years old, familiar with computers and Web searching). Every participant had to submit each of these queries to a special evaluation system [8] that we developed which visualizes the results of the three clustering algorithms (STC, STC+, NM-STC) in parallel (we used the parameters $K = 100$, $LL_{min} = 1$, $LL_{max} = 4$, $NC_{max} = 15$). The users did not know which algorithms were used, and they were free to submit whatever query they liked. After inspecting the results, each participant had to rank the three methods according to (a) label readability, (b) cluster ordering, (c) number of clusters and (d) overall quality. In this way we collected 16 * 11 * 4 = 704 user assessments in total. The users expressed their preference by providing numbers from {1, 2, 3}: 1 to the best, and 3 to the worst. Ties were allowed, e.g. STC:1, STC+:1, NM-STC:2 means that the first two are equally good, and NM-STC is the worst. In case all three were indifferent (they liked/disliked them equally), they were giving the value 0. We aggregated the rankings using Plurality Ranking (PR) (i.e. by considering only the winners) and Borda Ranking (BR) [5]. The middle part of Table 1 reports the average results[9]. In the PR column, the higher a

[6] Developed by the Dep. of Computer Science (U. of Crete) and FORTH-ICS.

[7] for more see: http://groogle.csd.uoc.gr:8080/mitos/files/clusteringEvaluation/UserEval.xls

[8] http://groogle.csd.uoc.gr:8080/clusteringEvaluation/, select Advanced Search, Results options: Clustering

[9] The PR value was computed by summing all ones (i.e. first positions) and then dividing by 11*16 (i.e. $|users| \times |queries|$)

value is, the better, while in the BR column the less a value is, the better. The rightmost part of Table 1 shows the relative ranking of the algorithms: 1 for the best, 2 for the second, and 3 for the third in preference algorithm (according to PR and BR). Notice that the relative ordering is the same for both PR and BR. The results show STC+ and NM-STC are clearly the most preferred algorithms according to each of the three criteria, and according to the overall assessment. In particular, NM-STC yields the more readable labels, STC+ yields the best cluster label ordering and NM-STC yields the best number of clusters. Regarding criterion (d) (overall quality), STC+ obtained the best result (PR: 7.08), NM-STC a slightly lower (PR: 6.91), while STC a much lower value (PR: 3.41).

Table 1. Comparative Evaluation by Users

Criterion	STC		STC+		NM-STC		STC		STC+		NM-STC	
	PR	BR	PR	BR	PR	BR	PR	BR	PR	BR	PR	BR
(a) Label Readability	2.41	33.5	6.25	23.16	9.41	20.83	3	3	2	2	**1**	**1**
(b) Cluster Ordering	4.75	28.33	7.33	21.75	6.41	24.9	3	3	**1**	**1**	2	2
(c) Number of clusters	2.33	33.5	5.83	23.33	10.41	19.91	3	3	2	2	**1**	**1**
(d) Best method (overall)	3.41	31.08	7.08	21.75	6.91	23.5	3	3	**1**	**1**	2	2

In addition, we asked the participants to answer a small questionnaire. Table 2 shows the questions and the answers received. The results show that the majority prefers (a) hierarchically organized labels, (b) labels comprising one to three words, and (c) 10-15 clusters.

Table 2. Questionnaire

Question	Results
Do you prefer Flat or Hierarchical cluster labels?	Flat (24%), Hierarchical (58%), Both are fine (18%)
Preferred cluster label length	$1-3$(75%) $3-6$(25%)
Preferred number of clusters	< 10 (25%) $10-15$ (62.5%) $15-20$ (12.5%)

Clustering Evaluation Metrics We conducted an additional comparative evaluation between STC, STC+, and NM-STC. We used the metrics defined in Table 3. B denotes the set of the labels returned by a clustering algorithm, and for a $p \in B$ we use $C(p)$ to denote the set of objects that are assigned to cluster label p by the clustering algorithm.

Coverage measures the degree that clusters' extensions cover the answer A (the closer to 1, the better the clusters "cover" the answer A). Its value is low if the clusters cover a small portion of A and this implies that the clusters do not summarize the entire contents of A. The *label precision* of a label p is the percentage of objects in the extension of p that contain all words of p. It is clear that the label precision of NM-STC is (by construction) always 1, but this is not true for the other STC-based algorithms (due to the base cluster merging).

Table 3 reports the average values for the queries used in the empirical evaluation. The overlap for NM-STC is computed over the maximal elements of B (i.e. those in $maximal_{\prec}(B)$). The results show that STC and STC+ have exactly the same coverage while NM-STC has slightly lower[10]. STC+ and NM-STC give smaller names than STC. STC+ and NM-STC have higher overlap (which is not bad). The label precision of STC+ is smaller than that of STC due to the threshold 0.4 vs 0.5 in base cluster merging. For threshold=0.3 the label precision of STC+ drops to 0.60 while for threshold=0.2 it further drops to 0.47. These results motivate the reason for not further decreasing this threshold.

Table 3. Evaluation Metrics and Results

Name	Definition	STC	STC+	NM-STC
coverage	$coverage = \frac{\lvert\cup_{p\in B} C(p)\rvert}{\lvert A\rvert}$	0.994	0.994	0.869
average label length	$LL_{avg} = avg_{p\in B}\lvert w(p)\rvert$	3.185	2.906	2.249
overlap	$AvO = \frac{2}{\lvert B\rvert(\lvert B\rvert-1)} \sum_{i=1}^{\lvert B\rvert} \sum_{j=i+1}^{\lvert B\rvert} JO(p_i, p_j)$ where $JO(p_i, p_j) = \frac{\lvert C(p_i)\cap C(p_j)\rvert}{\lvert C(p_i)\cup C(p_j)\rvert}$	0.038	0.048	0.099
label precision	$AvLP = \frac{1}{\lvert B\rvert} \sum_{p\in B} LabelPrec(p)$ where $LabelPrec(p) = \frac{\lvert\{o\in C(p) \mid w(p)\subseteq w(o)\}\rvert}{\lvert C(p)\rvert}$	0.893	0.756	1.0

Time Performance For the evaluation queries we counted the average time to cluster the top-100, the top-200 and the top-300 snippets. In NM-STC the IDF of the terms are in main memory from the beginning. Also recall that PIDF could be omitted from the scoring formula as it does not seem to influence the results (except in cases of very small result sets). The measured times (in seconds) are shown next (using a Pentium IV 4 GHz, 2 GB RAM, Linux Debian).

Alg	Top-100	Top-200	Top-300
STC	0.208	0.698	1.450
STC+	0.228	0.761	1.602
NM-STC	0.128	0.269	0.426

Notice that NM-STC is (two to three times) faster than STC and STC+. This is because NM-STC does not have to intersect and merge base clusters.

6 Conclusion and Future Work

In this work we focused on suffix tree clustering algorithms because they are fast, they do not rely on external resources or training data, and thus they have broad applicability (e.g. different natural languages). We presented a variation of the STC, called STC+, with a scoring formula that favors phrases that occur in document titles, and a novel suffix tree based algorithm called NM-STC that results in hierarchically organized clusters. The advantages of NM-STC are that:

[10] In general all coverage values are acceptably high, e.g. higher than those in [12], and by adding an artificial "rest" cluster label we could achieve 100% coverage.

(a) the user never gets unexpected results, as opposed to the existing STC-based algorithms which adopt overlap-based cluster merging, (b) it is more configurable w.r.t. desired cluster label lengths (STC favors specific lengths), (c) it derives hierarchically organized labels, and (d) it favors occurrences in titles (as STC+) and takes into account IDFs, if available. The user evaluation showed that both STC+ and NM-STC are significantly more preferred than STC (STC+ is slightly more preferred than NM-STC). In addition NM-STC is about two times faster than STC and STC+. In future we plan to work towards further improving the quality of cluster labels and the interaction with the user.

References

1. H. Andreka, M. Ryan, and P.-Y. Schobbens. Operators and Laws for Combining Preference Relations. *Journal of Logic and Computation*, 12(1):13–53, 2002.
2. D. Crabtree, X. Gao, and P. Andreae. Improving web clustering by cluster selection. In *Procs of the IEEE/WIC/ACM Intern. Conf. on Web Intelligence (WI'05)*, pages 172–178, Compiegne, France, September 2005.
3. D.R. Cutting, D. Karger, J.O. Pedersen, and J.W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Procs of the 15th Annual Intern. ACM Conf. on Research and Development in Information Retrieval, (SIGIR'92)*, pages 318–329, Copenhagen, Denmark, June 1992.
4. W. Dakka and P.G. Ipeirotis. Automatic extraction of useful facet hierarchies from text databases. In *Procs of the 24th Intern. Conf. on Data Engineering, (ICDE'08)*, pages 466–475, Cancún, México, April 2008.
5. J. C. de Borda. Memoire sur les Elections au Scrutin, 1781. Histoire de l'Academie Royale des Sciences, Paris.
6. P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Procs of the 14th Intern. Conf. on World Wide Web, (WWW'05)*, volume 5, pages 801–810, Chiba, Japan, May 2005.
7. B.C.M. Fung, K. Wang, and M. Ester. Hierarchical Document Clustering Using Frequent Itemsets. In *Procs of the SIAM Intern. Conf. on Data Mining*, volume 30, San Francisco, CA, USA, May 2003.
8. F. Gelgi, H. Davulcu, and S. Vadrevu. Term ranking for clustering web search results. In *10th Intern. Workshop on the Web and Databases, (WebDB'07)*, Beijing, China, June 2007.
9. D. Gusfield. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. *ACM SIGACT News*, 28(4):41–60, 1997.
10. M.A. Hearst and J.O. Pedersen. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *Procs of the 19th Annual Intern. ACM Conf. on Research and Development in Information Retrieval, (SIGIR'96)*, pages 76–84, Zurich, Switzerland, August 1996.
11. J. Janruang and W. Kreesuradej. A new web search result clustering based on true common phrase label discovery. In *Procs of the Intern. Conf. on Computational Intelligence for Modelling Control and Automation and Intern. Conf. on Intelligent Agents Web Technologies and International Commerce, (CIMCA/IAWTIC'06)*, page 242, Washington, DC, USA, November 2006.
12. M. Käki. Findex: properties of two web search result categorizing algorithms. In *Procs of the IADIS Intern. Conf. on World Wide Web/Internet*, Lisbon, Portugal, October 2005.

13. P. Papadakos, S. Kopidaki, N. Armenatzoglou, and Y. Tzitzikas. Exploratory web searching with dynamic taxonomies and results clustering. In *Procs of the 13th European Conference on Digital Libraries, (ECDL'09)*, Corfu, Greece, September 2009.
14. P. Papadakos, Y. Theoharis, Y. Marketakis, N. Armenatzoglou, and Y. Tzitzikas. Mitos: Design and evaluation of a dbms-based web search engine. In *Procs of the 12th Pan-Hellenic Conf. on Informatics, (PCI'08)*, Greece, August 2008.
15. P. Papadakos, G. Vasiliadis, Y. Theoharis, N. Armenatzoglou, S. Kopidaki, Y. Marketakis, M. Daskalakis, K. Karamaroudis, G. Linardakis, G. Makrydakis, V. Papathanasiou, L. Sardis, P. Tsialiamanis, G. Troullinou, K. Vandikas, D. Velegrakis, and Y. Tzitzikas. The Anatomy of Mitos Web Search Engine. *CoRR, Information Retrieval*, abs/0803.2220, 2008. Available at http://arxiv.org/abs/0803.2220.
16. G. M. Sacco and Y. Tzitzikas (Editors). *Dynamic Taxonomies and Faceted Search: Theory, Practise and Experience*. Springer, 2009.
17. J. Stefanowski and D. Weiss. Carrot2 and language properties in web search results clustering. In *Procs of the Intern. Atlantic Web Intelligence Conf., (AWIC'03)*, Madrid, Spain, May 2003. Springer.
18. Y. Tzitzikas, N. Armenatzoglou, and P. Papadakos. FleXplorer: A Framework for Providing Faceted and Dynamic Taxonomy-Based Information Exploration. In *19th Intern. Workshop on Database and Expert Systems Applications, (FIND '08 at DEXA'08)*, pages 392–396, Torino, Italy, September 2008.
19. E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.
20. J. Wang, Y. Mo, B. Huang, J. Wen, and L. He. Web Search Results Clustering Based on a Novel Suffix Tree Structure. In *Procs of 5th Intern. Conf. on Autonomic and Trusted Computing, (ATC'08)*, volume 5060, pages 540–554, Oslo, Norway, June 2008.
21. Y. Wang and M. Kitsuregawa. Use link-based clustering to improve Web search results. In *Procs of the Second Intern. Conf. on Web Information System Engineering, (WISE'01)*, Kyoto, Japan, December 2001.
22. P. Weiner. Linear pattern matching algorithms. In *14th Annual Symposium on Foundations of Computer Science*, pages 1–11, USA, October 1973.
23. D. Weiss and J. Stefanowski. Web search results clustering in Polish: Experimental evaluation of Carrot. In *Procs of the International IIS: Intelligent Information Processing and Web Mining, (IIPWM'03)*, Zakopane, Poland, June 2003.
24. D. Xing, G.R. Xue, Q. Yang, and Y. Yu. Deep classifier: Automatically categorizing search results into large-scale hierarchies. In *Procs of the Intern. Conf. on Web Search and Web Data Mining, (WSDM'08)*, pages 139–148, Palo Alto, California, USA, February 2008.
25. O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Procs of the 21th Annual Intern. ACM Conf. on Research and Development in Information Retrieval, (SIGIR'98)*, pages 46–54, Melbourne, Australia, August 1998.
26. O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. *Computer Networks*, 31(11-16):1361–1374, 1999.
27. H.J. Zeng, Q.C. He, Z. Chen, W.Y. Ma, and J. Ma. Learning to cluster web search results. In *Procs of the 27th Annual Intern. Conf. on Research and Development in Information Retrieval, (SIGIR'04)*, pages 210–217, Sheffield, UK, July 2004.
28. D. Zhang and Y. Dong. Semantic, Hierarchical, Online Clustering of Web Search Results. In *6th Asia-Pacific Web Conf. on Advanced Web Technologies and Applications, (APWeb'04)*, pages 69–78, Hangzhou, China, April 2004.