Web Intelligence and Agent Systems: An International Journal 0 (2010) 1–0
IOS Press

Full discrimination of subtopics in search results with keyphrase-based clustering [1]

Claudio Carpineto , Massimiliano D'Amico and Andrea Bernardini
Fondazione Ugo Bordoni,
Via Baldassarre Castiglione 59, I-00142 Rome, Italy
E-mail: carpinet@fub.it, mas.damico@gmail.com, aberna@fub.it

Abstract. We consider the problem of retrieving multiple documents relevant to the single subtopics of a given web query, termed "full-subtopic retrieval". To solve this problem we present a novel search results clustering algorithm that generates clusters labeled by keyphrases. The keyphrases are extracted from the generalized suffix tree built from the search results and merged through an improved hierarchical agglomerative clustering procedure. Our approach has been implemented into KeySRC (Keyphrase-based Search Results Clustering), a full web clustering engine available online at http://keysrc.fub.it. We discuss how the keyphrase-based clustering algorithm can be used not only for browsing through the clustered search results but also for producing a re-ranked list of results emphasizing the diversity of top hits. Using a novel measure for evaluating full-subtopic retrieval performance, called "Subtopic Search Length under k document sufficiency", and a test collection specifically designed for evaluating subtopic retrieval, we found that our approach was able to discriminate between the different subtopics present in search results in a very effective manner, with a clear improvement over other subtopic retrieval systems. In particular, browsing through KeySRC clusters was the best method to retrieve more documents per subtopic (i.e., $k > 1$), whereas using the re-ranked list formed from KeySRC clusters was more effecive for retrieving just one document per subtopic (i.e., $k = 1$).

Keywords: Search results clustering, subtopic retrieval, retrieval performance evaluation, generalized suffix tree, search results diversification

1. Introduction

While search engines are definitely good for certain search tasks such as finding the home page of an organization, they may be less effective for satisfying broad or ambiguous informational queries. The results on different subtopics of a query will be typically mixed together in the ranked list returned by the search engine, thus implying that the user may have to sift through a large number of irrelevant items to locate those of interest. The number of real user queries affected by this problem is potentially large, partly because informational queries have been estimated to account for 80% of web queries [17], and partly because today virtually any web query expressed by very few words has multiple subtopics (or meanings, or interpretations).

One approach is to try to optimize the relevance of results at the document set level instead of at the single document level. This is done by re-ranking the list of search results with the goal of delivering a first results page with reduced redundancy. The *diversification* of results is achieved by way of similarity functions that usually trade relevancy for novelty (e.g., [29], [31], [24], [30]). While this approach may be very useful for quickly finding at least one relevant subtopic document, it does not seem able to facilitate the retrieval of more information on specific subtopics of interest, which is often the primary goal of the efforts of searchers [27]. We refer to the retrieval of multiple documents relevant to a query's subtopic as *full-subtopic retrieval*.

The most natural way to address this issue is probably search results clustering (e.g., [28], [6], [21], [9],

[1] This paper is an extended version of [2].

[14]), which consists of partitioning the results obtained in response to a query into a set of labeled clusters that reflect the different query's subtopics.

As an illustration, in Figure 1 we show a screenshot of KeySRC (i.e., the search results clustering system that will be described in this paper), relative to the query "data mining". The clusters created by the system analyzing the first 98 results returned by a major search engine are displayed on the left-hand side. Each cluster is described by a label and a number specifying the number of documents contained in that cluster. As you can see, the cluster labels identify several useful aspects or subtopics of this broad query, such as "business intelligence", "data mining tools", "data mining conference", "data mining researchers", and others. By clicking on a cluster label, the user can see the search results grouped under that cluster. In Figure 1, we show the six results associated with the cluster "data mining conference" (displayed in red), all of which are about conferences in the data mining field.

In general, if the items that relate to a same subtopic have been correctly placed within the same cluster and if the user is able to choose the right cluster from the cluster labels, such items can be accessed in logarithmic rather than linear time. The quality of cluster labels is thus paramount to the success of these systems.

Full-subtopic retrieval poses not only challenges for developing effective algorithms but also for evaluating performance. Inspired by Cooper's search length [11], we introduce a new evaluation measure called *Subtopic Search Length under k document sufficiency* ($kSSL$). The idea is to consider the number of elements that the user must examine to retrieve a specified number (k) of documents relevant to the single subtopics of a query. The shorter the search length, the better the system performance.

We define $kSSL$ for both ranked lists and clustered results, thus facilitating cross comparison between systems that employ different search paradigms. Also, to make evaluation of search results clustering systems as realistic as possible, it is anticipated that the selection process on the part of the user be driven by the appropriateness of the cluster labels to the subtopic being searched.

To address full-subtopic retrieval in an effective manner we present a new search results clustering algorithm explicitly designed for producing highly expressive cluster labels. The algorithm is based on extracting and analyzing *keyphrases* contained in the search results, through a combination of natural language processing and clustering techniques. Candidate keyphrases are first generated through the Generalized Suffix Tree (GST) associated with the search results, then they are filtered by POS tagging and statistical criteria, and are finally merged through agglomerative hierarchical clustering. The final cluster labels produced by the system are detailed, linguistically well-formed descriptions of the cluster content that can be interpreted individually. The algorithm has been implemented into KeySRC, a full web clustering engine.

Clearly, KeySRC is intended as a means for users to browse through clustered search results, but it does more than this. The same clustering algorithm can be used to produce a re-ranked list of search results that promotes diversity early in the ranking, by emphasizing documents that belong to different clusters. The output in this case is a list rather than a set of clusters. In this paper we explore both KeySRC-based approaches to discriminating subtopics in search results, termed $KeySRC_C$ (for clustered results) and $KeySRC_L$ (for re-ranked list of results).

Using $kSSL$ and a test collection specifically designed for evaluating subtopic retrieval, we compared $KeySRC_C$ and $KeySRC_L$ to other state-of-the-art search results clustering algorithms and also to using the original list of search results or a re-ranked list based on subtopic coverage. In our experiments $KeySRC_C$ outperformed all other clustering algorithms for all values of k, and all list-based methods including $KeySRC_L$ for $k > 1$. For $k = 1$, the best result was achieved by $KeySRC_L$.

To summarize, this paper offers five main contributions:

- a new performance measure for evaluating subtopic retrieval,
- a new search results clustering algorithm,
- KeySRC, a web clustering engine in which the new algorithm has been implemented,
- a search results re-ranking method based on KeySRC,
- a comprehensive comparative evaluation of different subtopic retrieval systems.

The rest of the paper has the following structure. We first introduce the full-subtopic retrieval task and the $kSSL$ evaluation metric. Then we present the keyphrase-based search results clustering algorithm and its implementation into KeySRC, followed by the description of the re-ranking method based on KeySRC and a discussion of related systems for subtopic retrieval. We next describe our experimental evaluation of the retrieval effectiveness of KeySRC, in-

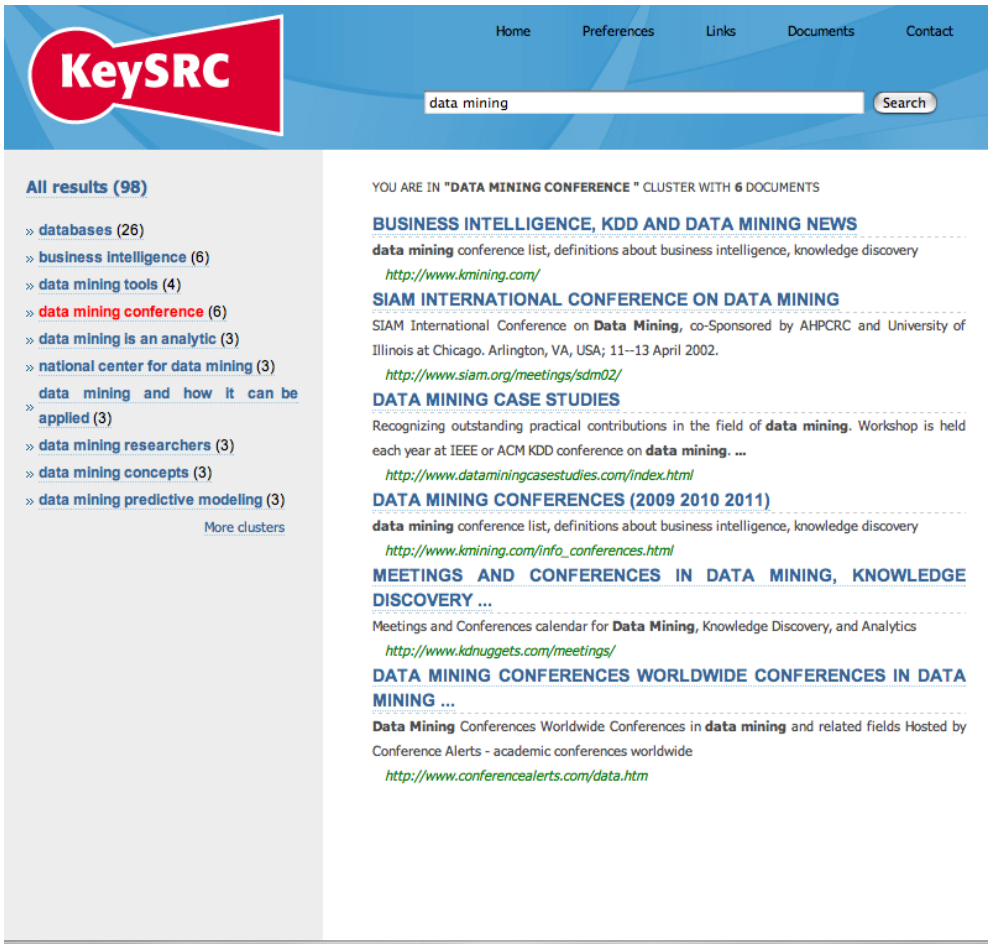

Fig. 1. Results of KeySRC for the query “data mining”.

cluding the systems used for comparison, the test collection, and the main findings of the experiments. The paper is completed by a discussion of limitations and future research directions and by some conclusions.

2. Full-subtopic retrieval: task definition and performance measure

For a given query, the input is represented by a set of web search results, each described by a URL, a title, and a snippet. We assume that the query spans a set of subtopics and that there is a subset of search results relevant to each subtopic. Subtopic subsets may overlap. The goal is to find an effective method to retrieve the full subset of documents relevant to any possible query’s subtopic.

The next step is the definition of a suitable performance evaluation measure. The *instance recall at n* [16], together with its refinements [29], is one of the few metrics centered on evaluating retrieval of query aspects, because it measures the percentage of different subtopics that are satisfied by the first n documents. Its focus is on subtopics rather than on documents: it does not assume that all subtopics must be retrieved by the system, but it cannot measure the ability of the system in retrieving more than one document per subtopic. Furthermore, it is only suitable for ranked lists. Here we define a measure for subtopic retrieval that has complementary features. It takes into account all the documents relevant to each query’s subtopic and can be applied to both ranked lists of results and to clustered results.

We define the *Subtopic Search Lenght under k document sufficiency* ($kSSL$) as the average number of search results that must be examined before finding a sufficient number (k) of documents relevant to any of the *n* query’s subtopics. For a ranked list, the value of

$kSSL$ is simply given by the mean of the ranks of the *k*-th result relevant to each subtopic in the ranked list associated with the query:

$$kSSL_{list} = \frac{\sum_{i=1}^{n} p_{i,k}}{n} \quad (1)$$

where $p_{i,k}$ is the rank of the *k*-th results relevant to subtopic *i*.

For clustered results, the definition of $kSSL$ is more complicated because we need to take into account both the cluster labels that must be scanned and the snippets that must be read. In addition, only the clusters with relevant labels to the subtopic at hand should be considered, because clusters with nonrelevant labels will be ignored by the user in the search process. This is to make sure that the evaluation is as realistic as possible – whereas most earlier studies assume that the user is able to select the clusters with relevant documents regardless of the cluster labels [5] – but in practice it requires that each cluster label be tagged as relevant or nonrelevant to the query's subtopic.

Our exact modelization of browsing through the clusters is the following. We assume that the user examines the cluster list top down, opening only the clusters with a relevant label to the subtopic of interest and examining their elements sequentially, until k relevant search results have been retrieved. The $kSSL$ value is the mean, averaged over the set of subtopics, of the number of pieces of information that must be examined for each subtopic according to this user behavior model.

For the case when it is sufficient to open only the first cluster (with a relevant label) to retrieve k relevant results, the value of $kSSL$ is given by:

$$kSSL_{clusters} = \frac{\sum_{i=1}^{n}(c_{i,k} + r_{i,k})}{n} \quad (2)$$

where $c_{i,k}$ is the rank of the cluster (considering their top-down order on the screen) containing the k relevant results to subtopic i, and $r_{i,k}$ is the rank of the *k*-th result in the cluster.

More generally, the contribution of each subtopic to the above formula is defined as the sum of three terms: the rank of the last of the m clusters with a relevant label that were visited before retrieving k results (denoted $c^*_{i,k}$), plus the sum of the cardinalities of the first $m-1$ visited clusters, plus the rank of the *k*-th relevant result in cluster $c^*_{i,k}$ (denoted $r^*_{i,k}$). Formally:

$$kSSL_{clusters} = \frac{\sum_{i=1}^{n}\left(c^*_{i,k} + \sum_{j=1}^{m-1}|c_{i,j}| + r^*_{i,k}\right)}{n} \quad (3)$$

If the list of clusters with a relevant label ends before retrieving k documents, it is necessary to switch to the full ranked list of documents and to add a further term to the sum equal to the number of search results that must be examined to retrieve the missing relevant documents.

It is worth noting that the minimum value of $kSSL$ depends on the number of subtopics. Topics with more subtopics have an inherently higher minimum $kSSL$, because the system will have to give access to more distinct tokens of information. Likewise, the minimum $kSSL$ grows as k increases. For instance, for k=1, $min\, kSSL_{list} = \frac{1+2+...+n}{n} = \frac{n+1}{2}$ (i.e., when each of the first n search results satisfies a distinct subtopic), and $min\, kSSL_{cluster} = \frac{2+3+...+(n+1)}{n} = \frac{n+3}{2}$, (i.e., when each of the first n clusters refers to a distinct subtopic and contains a relevant document in the first position). For k=2, $min\, kSSL_{list} = \frac{2+4+...+2n}{n} = n+1$, and $min\, kSSL_{cluster} = \frac{3+4+...+(n+1)}{n} = \frac{n+5}{2}$.

This behavior is perfectly reasonable, but it may result in an excessive penalization of the topics with few subtopics when the variability in the number of subtopics in the test collection is high. To make search lengths more comparable across topics, the individual values might be normalized over the number of subtopics. A more principled manner would be to normalize over the best theoretical case, following the approach taken in [29], but this is computationally very expensive. Notice also that while in our definition of $kSSL$ all subtopics have the same importance, we might compute a weighted average where each subtopic weight is proportional to the number of search results covered by that subtopic, thus emphasizing retrieval from popular subtopics.

3. Keyphrase-based search results clustering: method description

The input is a ranked list of 100 search results returned from a plain web search engine in response to a user query. The output is a set of possibly overlapping clusters in which the search results have been grouped. Our method consists of the following main steps:

1. *Search results pre-processing*. The search results go through standard processing stages for text extraction and normalization: text segmentation, word stemming, and stop wording. The two last operations are useful for improving the computational efficiency and reducing noise. Search results with a missing title/snippet are removed, which is why in the example in Figure 1 there are only 98 items.

2. *Construction of Generalized Suffix Tree (GST)*. The phrases contained in the search results can be extracted in time linear with the size of the input using a GST. For a sequence of elements $e_0, e_1, e_2, \ldots e_i$ a *suffix* is defined as a subsequence $e_j, e_{j+1}, e_{j+2}, \ldots e_i$, where $j \leq i$. A *suffix tree* for a given sequence of elements contains any of its suffixes on the path from root to a leaf node. A GST is similar to a suffix tree, but contains suffixes of more than one input sequence with internal nodes storing pointers to original sequences. We build the GST associated with the pre-processed search results using Andersson et al.'s algorithm [1], which is an improvement of Ukkonen's suffix tree construction algorithm [26] for the case when we are interested in suffixes that start at word boundaries.

3. *Extraction of keyphrases from GST*. From the huge set of phrases contained in the GST a much smaller set of keyphrases is selected according to the following criteria. Each phrase (a) must not be equal to the query, (b) must be contained in at least two search results (this is equivalent to considering only the internal nodes of the GST), (c) must contain no more than four words, (d) must contain only adjectives and nouns, including proper names. For computational reasons, criterion (d) has been implemented as a simple form of POS tagging, in which a valid word is either a word that has been defined as an adjective or noun in a large hashed morphological lexicon for English [18], or is a word that is not present in the lexicon.

4. *Keyphrase clustering*. Many keyphrases are different but refer to the same subtopic. In order to group the keyphrases with similar meaning, we represent each keyphrase as a document vector and perform hierarchical agglomerative clustering of the keyphrase vectors. The *i*-th position of each vector is given by the number of occurrences of the keyphrase in the *i*-th document divided by the log of the *i*-th document length, the similarity between keyphrase vectors is given by their normalized scalar product, and the similarity between clusters is computed with the group average method. For choosing the cut level of the clustering dendrogram, we do not use a fixed similarity threshold, because the characteristics of a query's subtopics may be highly variable. Rather, we rely on a variable threshold that is specific to any pair of clusters being merged and represents a fraction of the average weighted similarities of their elements. The value of the similarity threshold $SimThr$ associated with clusters c_1 and c_2 is given by:

$$SimThr = const \ \frac{ics(c_1)\,|c_1| + ics(c_2)\,|c_2|}{|c_1| + |c_2|} \quad (4)$$

where ics is the average intra-cluster similarity of a given cluster (i.e., the average similarity of all pairs of elements in the cluster) and $const$ is a constant that has been experimentally set to 0.8. In practice, cluster merging is allowed only if the similarity between two clusters exceeds the similarity threshold associated with the pair. The clustering process halts as soon as this constraint can no longer be satisfied, without the need for generating the entire dendrogram. Each keyphrase cluster is then associated with a set of documents, formed by the documents that contain at least one of those keyphrases.

5. *Label assignment*. We need to choose a label for the clusters that have been formed through the procedure described so far. This is a crucial step because a label is often not sufficiently complete to comprehend the meaning of documents in a cluster and/or it does not adequately reflect their composition. We select the keyphrase contained in a given cluster that maximizes both the intensional and extensional coverage of the cluster. This ensures that the keyphrase occurs in several documents and contains several discriminative terms. The score of a keyphrase $Score(KP)$ is given by:

$$Score(KP) \ = \ freq(KP) \sum_{w \in W_{KP}} freq(w) \quad (5)$$

where $freq(KP)$ is the number of search results in the cluster that contain KP, W_{KP} is the set of words contained in KP, and $freq(w)$ is the number of keyphrases in the cluster that contain word w. Notice that a cluster label does not necessarily appear, in its exact form, in each document covered by the cluster.

6. *Cluster ranking*. The dendrogram clusters along with their labels are thus ordered for top-down display, based on the cardinality of their set of search results (ties are broken with label scores). When all search search results have been covered, we discard

the remaining clusters to make the output cluster list more manageable. Finally, at the visualization stage, we reintroduce the stop words in the keyphrase labels to make sure that they are fully correct and meaningful from a linguistic point of view (unless the same keyphrase appeared with different stop words, in which case we do not expand it).

3.1. An example

To illustrate our method, we now give a detailed example. Consider the word "*zebra*", which is among other things a mammal, a type of mussel, and a free routing software. We show below a small subset of the documents (with a shortened snippet) that can be retrieved from a search engine in response to the query "*zebra*".

- *D1*: Harmful aquatic hitchikers: mollusks, zebra mussel.
- *D2*: Zebra mussel: name of a species of mollusks.
- *D3*: Zebra mussel originated in the Balkans, Poland.
- *D4*: Free routing software distributed under GNU license.
- *D5*: Zebra is open source TCP/IP routing software.
- *D6*: Zebra is the common name for some mammals of the genus equus.
- *D7*: Horselike African mammals of the genus equus.

The generalized suffix tree associated with the seven documents, after simple stop word removal, is shown in Figure 2. We use the following notations. Long suffixes are shortened by using the ".." symbol. A "$" indicates the document where the suffix occurs. For instance, the node "distributed..$4' (on the left in Figure 2) corresponds to the suffix "software distributed under GNU license", contained in document D4. Internal nodes correspond to portions of suffixes contained in at least two documents; e.g., the leftmost internal node "software" is a pointer to the two suffixes "software" (in D5) and "software distributed under GNU license" (in D4). Note that in Figure 2 we did not show for space limitations the suffixes that would be directly linked to the root; e.g., the suffix "Free routing software distributed under GNU license" in document D4. In our example, each internal node but "zebra" is a valid keyphrase, according to the criteria described above.

Assuming that each keyphrase is described by a simple binary document vector (with no length normalization), the corresponding similarity matrix computed via cosine similarity is shown in Table 3.1. Note that

Table 1
Similarity matrix for the keyphrases extracted from the seven documents. A = software, B = routing software, C = genus equus, D = equus, E = mussel, F = zebra mussel, G = mollusk, H = mammal genus equus, I = name.

	A	B	C	D	E	F	G	H	I
A	1	1	0	0	0	0	0	0	0
B	1	1	0	0	0	0	0	0	0
C	0	0	1	1	0	0	0	1	0.5
D	0	0	1	1	0	0	0	1	0.5
E	0	0	0	0	1	1	0,81	0	0.41
F	0	0	0	0	1	1	0,81	0	0.41
G	0	0	0	0	0,81	0,81	1	0	0.5
H	0	0	1	1	0	0	0	1	0.5
I	0	0	0	0.5	0.5	0.41	0.41	0.5	1

there are many 1's because several keyphrases are contained in exactly the same documents.

The dendrogram of the group-average hierarchical agglomerative clustering of the keyphrases is shown in Figure 3. On the Y axis we show the clusters created at each iteration (note that in this example the similarity value for the first four mergings is always equal to 1). The algorithm halts after five iterations, when it is no longer possible to merge two remaining clusters such that their similarity is above their dynamic similarity threshold (see dashed lines in Figure 3).

At this point, for each of the four resulting clusters, the keyphrase with the best coverage score according to Equation 5 is selected as label. For instance, for cluster {software, routing software} and its associated set of documents {D4, D5}, the keyphrase 'software" gets a value of 4, while the score of 'routing software" is 6. The other coverage scores are the following. For cluster (D6, D7): mammals genus equus (12), genus equus (10), equus (6); for cluster (D1, D2, D3): mussel (6), zebra mussel (9), mollusks (2); for cluster (D2, D6): name (2).

The ranked order of the four labeled clusters is: (D1, D2, D3), (D6, D7), (D4, D5), (D2, D6), after which the last cluster is discarded because all the documents have been covered by the preceding three clusters. The final ranked list of clusters output by the algorithm, after reintegrating the suppressed stop words, is:

- zebra mussel (D1, D2, D3)
- mammals of the genus Equus (D6, D7)
- routing software (D4, D5)

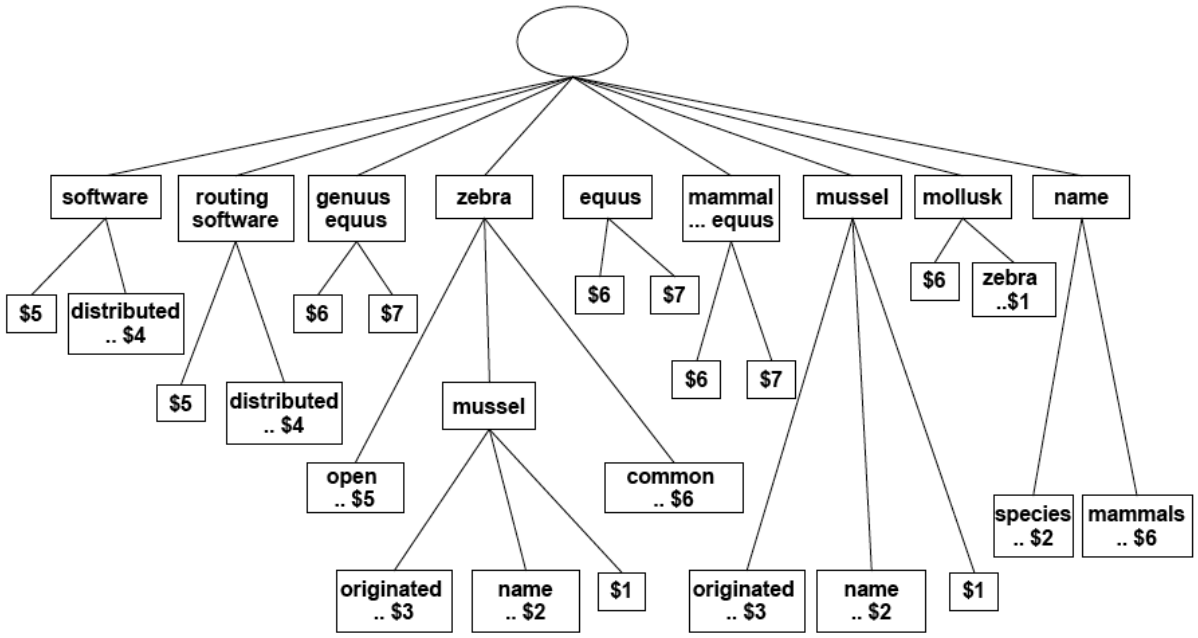

Fig. 2. Generalized suffix tree for the seven example documents (leaf nodes directly linked to the root are not shown).

3.2. System implementation

The algorithm described above is the core component of KeySRC (Keyphrase-based Search Results Clustering), a full web clustering engine available online at http://keysrc.fub.it. KeySRC collects the search results from Yahoo! (we did not use Google due to its restrictive constraints and term policy), clusters them and offers the clustered results to the user. The computational time complexity of the algorithm described above is quadratic with respect to the number of keyphrases to be clustered, due to the agglomerative hierarchical clustering step. In practice, however, the time necessary to do the clustering is negligible because only 100 search results per query are clustered. In fact, the most critical factor for the overall system's efficiency is search results acquisition, because the 100 search results cannot be fetched in one remote request. The response time of KeySRC is of the order of a couple of seconds, almost entirely due to the search results acquisition phase.

4. Using KeySRC to re-rank search results

Assuming that documents grouped in a same cluster are relevant to the same subtopic and nonrelevant to the other subtopics, a possible approach to increasing diversity in early ranking is to promote documents belonging to different clusters.

We selected a representative document from each cluster generated by KeySRC and added it to the ranking list of documents. The list of clusters was visited top-down and the representative was the most relevant document in the cluster according to the ranking function used by the search engine that retrieved the original list of documents. Note that in principle this method could be iterated, choosing in each round the first of the remaining documents in any cluster. However, as in our application there are many clusters with very few documents, we decided to perform only one round.

A different way of selecting the best representative in each cluster is to use the medoid, namely the document closest to the centroid of the cluster. The distance between two documents might be computed using a (dis)similarity function based on the set of terms describing the documents, but in our case this is probably not very appropriate because the documents have been clustered using transformed features (i.e., keyphrases) that do not always occur in the same form in the original representation of documents.

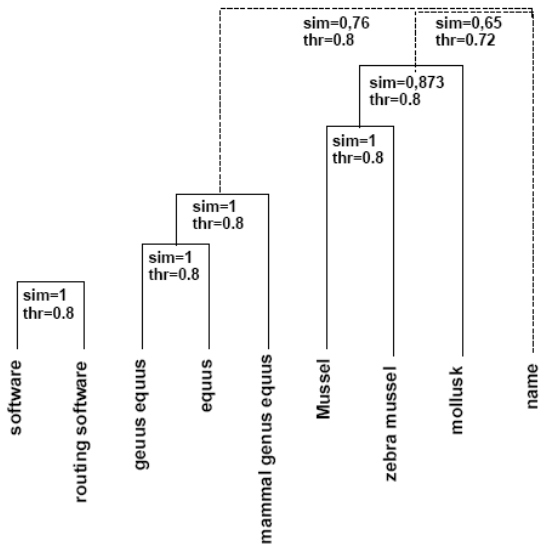

Fig. 3. Group-average agglomerative hierarchical clustering of keyphrases. We show the pair of clusters merged at each iteration, with the corresponding similarity and threshold values computed from the similarity matrix in Table 3.1.

5. Related work

Lists of subtopics for refining web searches were provided by AltaVista's Refine Feature as early as 1997, but it was only recently that they have become a popular search method, both for analyzing search results or formulating a better query. In the last few years, many new search results clustering algorithms optimized for usability have been proposed and several research and commercial web clustering engines have been deployed, the best known of which is probably Vivísimo (see [5] for a survey). In the shift from data-centric to description-centric algorithms, a variety of clustering paradigms have been used, including singular value decomposition [21], concept lattices [6], spectral clustering [9], and graph theory [14].

Most closely related to our work are the systems based on GST. After Suffix Tree Clustering (STC) [28], which was the first system to employ GST but was limited by its chaining mechanism for aggregating GST nodes, two main approaches have been followed: using the GST nodes for document indexing and then using these representations for measuring document similarity while clustering ([23], [10]), or selecting the most informative GST nodes and then building the clusters around them ([12], [22]). KeySRC belongs to the latter research line but it has several unique features, such as using POS for GST pruning, employing a dynamic cut-off level of the clustering dendrogram, and choosing the labels that maximize the intensional and extensional cluster coverage. Furthermore, earlier algorithms based on GST may not be computationally suitable for on-line web search.

Promoting diversity in the top returned documents is another approach to subtopic retrieval. Among the approaches that generate a set of diversified results, we are most interested in those which explicitly model the variance of subtopics in groups of documents, rather than those which try to optimize the relevance of the set of documents to the whole query as in [8]. Most relevant to this paper is Essential Pages (EP) [24], which optimizes for diversity by selecting a set of documents with maximum coverage of the web knowledge on a given query. Unlike most other search results re-ranking systems (e.g., [31], [30]), it does not require external knowledge or training documents and it can thus be easily applied to web search. We now describe it in more detail.

In EP, the joint coverage score of a set of documents is expressed as a function of the coverage score of their constituent words. The coverage score of a single word is computed using the *entropy* of the relevance of that word to the query: words that have low or high relevance have little coverage, the remaining words are deemed important. Finally, the relevance of a word to the query is estimated by dividing the number of documents containing that word *and* at least one query term by the total number of documents containing at least one query term. Thus, only words appearing together with query terms have non-zero relevance and contribute to the coverage score of the documents that contain them.

Although EP has been devised for working with full documents, it is suggested that for reducing noise only the paragraphs containing the query words should be considered. Thus, using search results snippets can be seen as an approximation of this recommended preprocessing step. Indeed, the authors of EP state that its recommended use is as post-processing tool of the results returned by a search engine.

Finally, we would like to point out that there has been some work on keyphrase extraction, with an emphasis on machine learning [25]. The specific requirements posed by our application prevented us from using more inefficient, albeit potentially more precise, methods such those requiring training documents with known keyphrases or knowledge intensive natural language processing techniques. On the other hand, some

very recent results suggest that keyphrase-extraction algorithms based on a combination of n-gram filtration, lexical analysis and word frequency counts may be equally effective [19].

6. Experimental evaluation

Using the $kSSL$ metric, we performed a comparative evaluation of the subtopic retrieval effectiveness of KeySRC. We now describe the main steps of our experiments, namely the systems that were evaluated, the test collection used, and the performance results.

6.1. Tested systems

Besides $KeySRC_C$ and $KeySRC_L$, we tested the five following systems.

- *STC*. Suffix Tree Clustering (STC) [28] has been described in the preceding section. In our experiments we used the version implemented in the open source Carrot[2] framework, available at http://project.carrot2.org.
- *Lingo*. Lingo [21] is a well known successor of STC. Frequent phrases are extracted using suffix arrays [20] instead of suffix trees, then the frequent phrases that best match certain *latent topics* present in the search results, extracted via singular value decomposition, are selected [13], and finally documents are allocated to such frequent phrases. Lingo can be tested at http://search.carrot2.org/stable/search.
- *CREDO*. CREDO [6] is based on the theory of concept lattices. The taxonomy of topics and their descriptions are built at once, exploiting the duality of *extents* (i.e., the documents containing certain terms) and *intents* (i.e., the terms contained in certain documents) in formal concepts. The system is available on line at http://credo.fub.it, and there are also mobile versions for PDAs [3] and cell phones [4].
- *EP*. Essential Pages (EP) [24] has been described in the preceding section.
- *EP**. Because we were interested in retrieving multiple documents per subtopic, we implemented a variant of EP (denoted EP*) aimed to choose multiple subsets of diverse documents, not just one subset of ten documents. In practice, we recursively applied the basic EP algorithm to the subsets of documents in the original ranking not chosen by earlier applications.
- *Yahoo!*. This was exactly the list of results given as an input to all other systems and it was used as a reference of comparison.

6.2. Test collection

There is no standard test collection for evaluating subtopic retrieval. Most earlier studies rely on the TREC 6-8 Interactive Track data, although this collection is focused on finding the *instances* of a given query (e.g., what tropical storms have caused property damage and/or loss of life), which is quite a distinct task from retrieving the subtopics (or the distinct meanings) of a query. Furthermore, the problem of reordering (or clustering) the search results is not decoupled from their retrieval, which makes it difficult to compare systems that only do post-processing of web search results.

We thus decided to build our own test collection, termed AMBIENT (AMBIguous ENTries). In summary, it consists of 44 topics extracted from the *ambiguous* Wikipedia entries, each with a set of subtopics and a list of 100 ranked search results annotated with relevance judgments per subtopic. For example, query 10 is 'Excalibur', and there are 8 retrieved subtopics in the set of search results obtained in response to the query 'Excalibur' (out of the 26 Wikipedia subtopics), with 32 search results that are relevant to at least one subtopic.[1] AMBIENT is fully described in [4] and is available at http://credo.fub.it/ambient for re-use by other researchers.

6.3. Results

The four clustering systems tested in our experiments (i.e., $KeySRC_C$, STC, Lingo, and CREDO) were run on the 100 search results associated with each AMBIENT query. Then, the relevance of each cluster label (produced by any system for any query) to any of the subtopics listed in AMBIENT for each query was manually assessed by three external subjects using a web-based evaluation tool developed for that purpose. As an illustration, consider again the query 'Excalibur'. We show in Table 2 the set of cluster labels gen-

[1] Here we used the term 'subtopic' to indicate the distinct meanings (or interpretations) of an ambiguous query. The same term can be used, perhaps in a more appropriate manner,for broad queries such as 'artificial intelligence' or 'data mining' (as in our introductory example). A recent test collection focusing on truly multi-topic queries is presented in [7].

Table 2

Cluster labels relevant to the AMBIENT subtopics for the query 'Excalibur'. The subtopic definitions are the following. Subtop1: 'Excalibur is the mythical sword of King Arthur'; Subtop2: 'Excalibur (film), a 1981 film about the legend of King Arthur'; Subtop3: 'Excalibur (automobile), a type of contemporary classic retro-styled car'; Subtop4: 'Excalibur Hotel and Casino, a medieval-themed hotel-casino in Las Vegas, Nevada'; Subtop5: 'XM982 Excalibur, 155mm extended range artillery projectile being developed by Raytheon and Bofors'

Excalibur qry	KeySRC	STC	Lingo	CREDO
Subtop1	king arthur's sword / magical sword / history of excalibur	Sword / Sword of Kings / Arthurian Legend / Magical Sword	Sword of King Arthur	arthur / sword / legend / camelot
Subtop2	excalibur on imdb movies / paul geoffrey nicol williamson	Movie	IMDb / DVD Reviews Excalibur	movie
Subtop3			Camelot Classic Car	
Subtop4	las vegas hotel	Las Vegas	Las Vegas Hotel	las vegas hotel
Subtop5	artillery projectile			artillery projectile

Table 3

Performance of subtopic retrieval systems on AMBIENT.

	kSSL (k=1)	kSSL (k=2)	kSSL (k=3)	kSSL (k=4)
$KeySRC_C$	14,40	**24,33**	**31,69**	**36,84**
STC	19,78	33,38	41,20	47,16
Lingo	15,79	27,06	36,012	40,67
CREDO	14.99	27.13	33.39	37.57
$KeySRC_L$	**11.43**	31.02	40.88	48.01
EP	16.09	31.27	40.25	47.32
EP*	13,79	25,83	33,21	38,04
Yahoo!	14,18	31,58	40,78	48,12

erated by each clustering algorithm that were judged to be relevant to at least one of the subtopics of 'Excalibur' defined in the AMBIENT collection.

We also computed the re-ranked list of search results using $KeySRC_L$, EP, and EP*. The performance of the outputs of the eight systems, including Yahoo!'s original search results list, was evaluated using the methodology described in Section 2. The $kSSL$ values are shown in Table 3, for several values of k (best results in bold).

For $k = 1$, the best result was achieved by $KeySRC_L$, followed by EP* and Yahoo!. This finding confirms the effectiveness of methods that promote diversity of results when we are interested in retrieving at least one search result per subtopic. The very good result of $KeySRC_L$ suggests that the first clusters generated by KeySRC actually refer to distinct subtopics, and that the first document in each such cluster is usually relevant. The good result of Yahoo! is not surprising because there are speculations that the major search engines have started to address the diversity issue in their first results page.

The superior performance of ranked lists over clustering for $k = 1$ can be also explained by considering the properties of our evaluation measure. For $k = 1$, the minimum $kSSL$ is equal to $\frac{n+1}{2}$ in the case of ranked list and equal to $\frac{n+3}{2}$ in the case of clustered results. The latter value is higher because the theoretically minimum cognitive effort involved with browsing clustered results is inherently greater than that required for browsing ranked lists. Using clustered results, the user will always have to scan both the list of clusters and the list of results within a selected cluster, whereas with a ranked list it may be sufficient to inspect just the first *n* results.

For $k > 1$, the situation was quite different. $KeySRC_C$ obtained the best results for any value of k compared to both ranked lists and clustering systems. The improvement over Yahoo! was overwhelming, with a reduction in search length of the order of 30% for any value of k. The superiority of $KeySRC_C$ over the other clustering systems was also clear, with dramatic improvements over STC and more limited but significant gains over CREDO and Lingo. $KeySRC_C$ outperformed the other clustering systems not only for $k > 1$ but also for $k = 1$, although by a smaller margin. The performance of EP* was very good overall, only slightly inferior to $KeySRC_C$ and clearly better than Yahoo!, whereas the results of the basic EP for $k > 1$ were much more similar to those of Yahoo!. EP* was better than EP* even for $k = 1$, which suggests

Table 4
Relationships between cluster labels and subtopics.

	single-topic labels (%)	covered subtopics (%)
KeySRC	51	65
STC	35	39
Lingo	42	50
CREDO	40	62

that some subtopics that were still low-ranked after the first re-ranking were promoted during the second re-ranking.

Table 3 also shows that the advantage of using clustering over the plain ranked list became more clear as the value of k increased. Even STC, the worst clustering system, eventually outperformed Yahoo! while all other clustering systems were consistently better than Yahoo!.

A final remark about the results in Table 3 is that the increase of $kSSL$ values seems to reduce proportionally as k grows. This phenomenon was due to the presence of some subtopics with very few relevant documents, i.e., there were many subtopics with three or just two relevant documents (note that all the subtopics used in our experiments had at least two relevant documents). If a subtopic had $j < k$ relevant documents, its contribution to $kSSL$ remained the same when passing from j to $j + 1, j + 2, ...k$, thus making retrieval of more documents apparently less difficult than expected. This is why we limited the maximum value of k considered in the experiments to 4.

The $kSSL$ metric used in the experiments described so far is very natural, because it integrates cluster labels and cluster structure into a single measure directly related to subtopic retrieval, which is the intended use of the performance clustering system. On the other hand, as one of our main goals in developing KeySRC was to produce more meaningful and expressive labels, it is also interesting to evaluate the quality of cluster labels per se, regardless of the clustering structure and the retrieval task. To this end, we counted the percentage of labels that were deemed appropriate to exactly one subtopic and the percentage of subtopics covered by at least one label. The results, shown in Table 4, suggest that the labels produced by KeySRC were both more meaningful and less ambiguous than those of the other systems, ensuring a wider coverage of subtopics in a more precise manner.

7. Discussion

Although search results clustering algorithms have significantly improved over the last few years, they still present several open problems, especially regarding the comprehensibility, granularity, completeness, and consistency of cluster labels [15]. For instance, one cluster label generated by KeySRC in response to the query ‘data mining’ (see Figure 1) was ‘data mining is an analytic’, which leaves much to be desired from the point of view of comprehensibility (although this problem could be remedied by using more powerful natural language processing tools). Or, taking another example, in the query ‘Excalibur’ used in our experiments (see Table 2), the labels generated for the Excalibur film varied from broad concepts such as ‘movie’ to specific aspects such as the names of two actors (i.e., ‘paul geoffrey nicol williamson’), or the reviews of the film on a specific web forum (i.e., ‘DVD Reviews Excalibur’). In addition, some systems identified certain interesting subtopic covering very few documents and other systems identified different subtopics of the same kind, but there was no single system covering all of them (e.g., ‘artillery projectile’ and ‘Camelot Classic Car’).

In spite of these limitations, clustering of search results is generally seen as a useful complement to plain search engines when the latter systems fail [5], e.g., for broad or ambiguous queries, or for mobile searches [4]. Our research reinforces this view. Full discrimination of subtopics in search results is a strong case in point, because plain search engines are very poor at this search task. We showed that browsing though clustered results may dramatically improve the retrieval effectiveness over just scanning the original list. Among several systems, KeySRC achieved the best retrieval performance, due to its keyphrase cluster labels and overall cluster structure.

Another interesting feature of search results clustering algorithms is that they can readily identify emerging trends, products, and services, or novel uses associated with a given term or concept. For instance, some of the subtopics of ‘Excalibur’ discovered by the search results clustering systems tested in our experiments were ‘Excalibur Fireplace Products’, ‘HTC Excalibur’, and ‘Excalibur cigars’, all of which are not in the list of Wikipedia subtopics of ‘Excalibur’.

There may be several follows-up on this research. A natural direction is to try to refine the quality and significance of keyphrases by using more powerful natural language techniques or external web data. The

process of cluster formation can be improved using various techniques such as personalization, integration with ontologies, and method combination. In addition, it would be interesting to make a systematic experimental comparison of clustering and diversification techniques, building on our preliminary results on their relative performance. Both clustering and diversification of search results have received a lot of attention recently and they seem to have complementary features, but there is still a lack of comparative evaluations.

8. Conclusion

We investigated the full-subtopic retrieval task by introducing a new performance metric ($kSSL$) that addresses some limitations of existing metrics for subtopic retrieval and provides a natural and realistic way of measuring the retrieval effectiveness of search results clustering systems. To solve this task more effectively, we presented a new search results clustering algorithm based on extraction and merging of keyphrases, and implemented it into the KeySRC web clustering engine. We used the output of KeySRC as a means for two search methods, namely browsing through clustered results ($KeySRC_C$) and re-ranking the original search list by promoting documents belonging to distinct clusters ($KeySRC_L$).

Through an experimental evaluation conducted using a suitable test collection and other state-of-the-art subtopic retrieval systems including clustering and diversification algorithms, we found that $KeySRC_C$ achieved the best $kSSL$ performance for retrieving more than one document relevant to a query's subtopic (i.e., for $k > 1$), while it was still better than other clustering systems but less effective than diversification, or even than plain search engines, when considering just one relevant document per subtopic (i.e., for $k = 1$). Of the search results re-ranking methods tested in our experiments, that based on KeySRC (i.e., $KeySRC_L$) achieved the best results for $k = 1$.

The main conclusions of this research are that search results clustering has the potential for effectively solving a search task, i.e.. full-subtopic retrieval, at which plain search engines clearly fail, and that keyphrase-based clustering is an efficient and perhaps more effective method than other description-centric clustering algorithms.

Acknowledgments

We would like to thank Stanislaw Osiński and Dawid Weiss for running Lingo and STC on the AMBIENT test collection and providing us with the results. Thanks also to Giovanni Romano for providing the analogous results of CREDO.

References

[1] A. Andersson, N. J. Larsson, and K. Swanson. Suffix trees on words. *Algorithmica*, 23:102–115, 1999.

[2] A. Bernardini, C. Carpineto, and M. D'Amico. Full-Subtopic Retrieval with Keyphrase-Based Search Results Clustering. In *Proceedings of Web Intelligence 2009, Milan, Italy*, pages 206–213. IEEE Computer Society, 2009.

[3] C. Carpineto, A. Della Pietra, S. Mizzaro, and G. Romano. Mobile Clustering Engine. In *Proceedings of the 28th European Conference on Information Retrieval, London, UK*, volume 3936 of *Lecture Notes in Computer Science*, pages 155–166. Springer, 2006.

[4] C. Carpineto, S. Mizzaro, G. Romano, and M. Snidero. Mobile Information Retrieval with Search Results Clustering: Prototypes and Evaluations. *JASIST*, 60(5):877–895, 2009.

[5] C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A survey of Web clustering engines. *ACM Computing Survey*, 41(3), 2009.

[6] C. Carpineto and G. Romano. *Concept Data Analysis — Theory and Applications*. Wiley, 2004.

[7] C. Carpineto and G. Romano. Optimal meta search results clustering. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval, Geneva, Switzerland*, pages 170–177. ACM Press, 2010.

[8] H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA*, pages 429–436. ACM Press, 2006.

[9] D. Cheng, S. Vempala, R. Kannan, and G. Wang. A divide-and-merge methodology for clustering. In C. Li, editor, *Proceedings of the 24th ACM Symposium on Principles of Database Systems, Baltimore, Maryland, USA*, pages 196–205. ACM Press, 2005.

[10] H. Chim and X. Deng. A new suffix tree similarity measure for document clustering. In *Proceedings of the 16th international conference on World Wide Web, Banff, Alberta, Canada*, pages 121–130. ACM Press, 2007.

[11] W. Cooper. Expected search length: A single measure of retrieval effectiveness based on weak ordering action of retrieval systems. *American Documentation*, 19(1):30–41, 1968.

[12] D. Crabtree, X. Gao, and P. Andreae. Improving web clustering by cluster selection. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, Compiegne University of Technology, France*, pages 172–178. IEEE, 2005.

[13] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and T. K. Harshman. Indexing by latent semanic analysis. *Journal*

of the American Society for Information Science, 41(6):391–407, 1990.

[14] E. Di Giacomo, W. Didimo, L. Grilli, and G. Liotta. Graph Visualization Techniques for Web Clustering Engines. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):294–304, 2007.

[15] M. A. Hearst. Clustering versus faceted categories for information exploration. *Communications of the ACM*, 49(4):59–61, 2006.

[16] W. R. Hersh and P. Over. TREC-8 Interactive Track Report. In *Proceedings of the 8th Text REtrieval Conference (TREC-8), Gaithersburg, Maryland, USA*. National Institute of Standards and Technology (NIST), 1999.

[17] B. J. Jansen, D. L. Booth, and A. Spink. Determining the informational, navigational, and transactional intent of Web queries. *Information Processing and Management*, 44(3):1251–1266, 2008.

[18] D. Karp, Y. Schabes, M. Zaidel, and D. Egedi. A freely available wide coverage morphological analyzer for English. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING '92), Nantes, France*, 1992.

[19] N. Kumar and K. Srinathan. Automatic keyphrase extraction from scientific documents using N-gram filtration technique. In *Proceedings of the 2nd European Semantic Web Conference, Heraklion, Greece*, pages 199–208. Springer, 2008.

[20] U. Manber and G. Myers. Suffix Arrays: A New Method for On-line String Searches. *SIAM Journal on Computing*, 22(5):935–948, 1993.

[21] S. Osiński and D. Weiss. A Concept-Driven Algorithm for Clustering Search Results. *IEEE Intelligent Systems*, 20(3):48–54, 2005.

[22] L. Ruixu and J. Whang. A new cluster merging algorithm of suffix tree clustering. In *FIP TC12 International Conference on Intelligent Information Processing (IIP 2006), Adelaide, Australia*, pages 197–203. Springer, 2006.

[23] S.Branson and A. Greenberg. Clustering Web search results using suffix tree methods. Technical Report CS276A Final Project, Stanford University, 2002.

[24] A. Swaminathan, C. Mathew, and D. Kirovski. Essential Pages. Technical Report MSR-TR-2008-15, Microsoft Research, 2008.

[25] P. D. Turney. Learning Algorithms for Keyphrase Extraction. *Information Retrieval*, 2(4):303–336, 2000.

[26] E. Ukkonen. On-Line Construction of Suffix Trees. *Algorithmica*, 14(3):249–260, 1995.

[27] Y. Xu and H. Yin. Novelty and topicality in interactive information retrieval. *Journal of the American Society for Information Science and Technology*, 59(2):201–215, 2008.

[28] O. Zamir and O. Etzioni. Web Document Clustering: A Feasibility Demonstration. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia*, pages 46–54. ACM Press, 1998.

[29] C. Zhai, W. W. Cohen, and J. Lafferty. Beyond Independent Relevance: Methods and Evaluation Metrics for Subtopic Retrieval. In *Proceedings of the 26th International ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada*, pages 10–17. ACM Press, 2003.

[30] C. Zhai, W. W. Cohen, and J. Lafferty. Diversifying search results. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining (WSDM 2009), Barcelona, Spain*, pages 5–14. ACM Press, 2009.

[31] B. Zhang, H. Li, Y. Liu, L. Ji, W. Xi, W. Fan, Z. Chen, and W.-Y. Ma. Improving web search results using affinity graph. In *Proceedings of the 28th International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil*, pages 504–511. ACM Press, 2006.