Full-Subtopic Retrieval with Keyphrase-based Search Results Clustering

Andrea Bernardini, Claudio Carpineto, Massimiliano D'Amico
Fondazione Ugo Bordoni
Rome, Italy
abernardini@fub.it, carpinet@fub.it, mas.damico@gmail.com

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. We also introduce a novel measure for evaluating full-subtopic retrieval performance, namely "Subtopic Search Length under k document sufficiency". Using a test collection specifically designed for evaluating subtopic retrieval, we found that our algorithm outperformed both other existing search results clustering algorithms and also a search results re-ranking method that emphasized diversity of results (at least for k>1; i.e., when we are interested in retrieving more than one relevant document per subtopic). Our approach has been implemented into KeySRC (Keyphrase-based Search Results Clustering), a full web clustering engine available online at http://keysrc.fub.it.

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 [1], 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 optimize the relevance of results at the document set level instead of at the single document level. This is usually 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 measure their novelty or coverage ([2], [3] [4], [5]). 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 [6]. 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., [7], [8], [9], [10], [11]), 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. 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 [12], 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 making cross comparison between systems that employ different search paradigms possible. 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.

We compare KeySRC to other state-of-the-art search results clustering algorithms using kSSL and a test collection specifically designed for evaluating subtopic retrieval. We also compare KeySRC to a search results re-ranking method that finds *essential pages* [4] with high subtopic coverage. In our experiment KeySRC outperforms the other clustering algorithm (for all values of k) and also the essential pages method (for all values of k, except for $k = 1$)

To summarize, this paper offers four 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 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 a discussion of related systems for subtopic retrieval. We next describe our experimental evaluation of the retrieval effectiveness of KeySRC, including the systems used for comparison, the test collection, and the main findings of the experiments. The paper is completed 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* [13], together with its refinements [2], is one of the few metrics centered on evaluating retrieval of query aspects, because it measures the number of different subtopics that are satisfied by the first n documents. However, it considers only one document per subtopic and it is only suitable for ranked lists. Here we define a comprehensive measure for subtopic retrieval that takes into account all the documents relevant to each query's subtopic and that can be applied to both ranked lists of results and also 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 positions of the *k*-th result relevant to each subtopic in the ranked list associated with the query:

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

where the index *k* refers to 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. Also, we should only consider the clusters with labels semantically appropriate to the subtopic at hand, because clusters with inappropriate 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 [14] – but in practice it requires that each cluster label be tagged as appropriate or inappropriate to the query's subtopic.

Our exact modelization of browsing through the clusters is the following. We assume that the user examines sequentially the cluster list, opens the first cluster with a semantically appropriate label, examines sequentially its elements, and then iterates this basic cycle moving to the next appropriate cluster in the list 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 just the first cluster (with an appropriate label) to retrieve k relevant results, the value of kSSL is given by:

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

where c_k is the position of the cluster (considering their top-down order on the screen) containing the k relevant results and r_k is the position 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 position of the last of the m clusters with an appropriate label (denoted c_k) that were visited before retrieving k results, plus the sum of the cardinalities of the first $m-1$ visited clusters, plus the position of the *k*-th relevant result in cluster c_k. If the list of appropriate clusters ends before retrieving k documents, it is necessary to switch to the full ranked list of documents and to add a further term equal to the number of search results that must be examined to retrieve the missing relevant documents.

It is worth noting that the minimum 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 $k = 1$, minimum $kSSL_{list} = \frac{n+1}{2}$ and minimum $kSSL_{cluster} = \frac{n+3}{2}$; for k=2, minimum $kSSL_{list} = n + 1$ and minimum $kSSL_{cluster} = \frac{n+5}{2}$. 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.

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.

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 [15], which is an improvement of Ukkonen's suffix tree construction algorithm [16] 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 by 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 [17], or it 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 dendogram, 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 threshold associated with clusters c_1 and c_2 is given by:

$$Thr = const \; \frac{ics(c_1)\,|c_1| + ics(c_2)\,|c_2|}{|c_1| + |c_2|}$$

where ics is the average intra-cluster similarity of a given 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 dendogram.

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 insufficient 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. The score of a keyphrase (KP) is given by:

$$Score \; = \; KPfreq \sum_{t \in W_{KP}} wfreq$$

where $KPfreq$ is the number of search results in the cluster that contain KP, W_{KP} is the set of words contained in KP, and $wfreq$ 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 dendogram 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.

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 1. 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 1. The dendogram of the group-average hierarchical agglomerative clustering of the keyphrases is shown in Figure 2. Notice that 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 threshold (see dashed lines in Figure 2). At this point, for each of the four resulting clusters, the keyphrase with best coverage is selected as label. The coverage scores are the following. For cluster (D4, D5): software (4), routing software (6); 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)

The algorithm described above is the core component of KeySRC (Keyphrase-based Search Results Clustering), a full web clustering engine available online. 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. As an illustration, in Figure 3 we show a screenshot of KeySRC relative to the query "artificial intelligence". The clusters created by

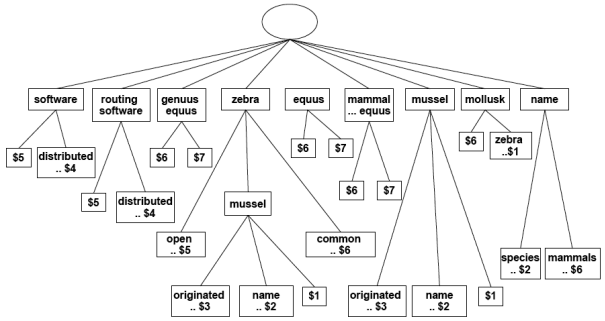

Figure 1. Generalized suffix tree for the seven documents.

	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

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.

the system show several useful aspects of this broad query, including the name of one of the fathers of the field and the title of a well known book. The response time of KeySRC is of the order of a couple of seconds, almost entirely due to the search results acquistion phase. KeySRC is available at http://keysrc.fub.it.

4. Related work

Following the Vivísimo's 'description comes first' motto, in the last years many new search results clustering algorithms optimized for usability have been proposed and several research and commercial web clustering engines have been deployed (see [14] for a survey). Although clustering engines are generally seen as a useful complement to plain search engines when the latter systems fail (e.g., for multi-topic or ambiguous querys) or for mobile searches [18], they still present several open problems, especially regarding the comprehensibility, granularity and consistency of cluster labels [19].

In the shift from data-centric to description-centric algorithms, a variety of clustering paradigms have been used, including singular value decomposition [9], concept lattices

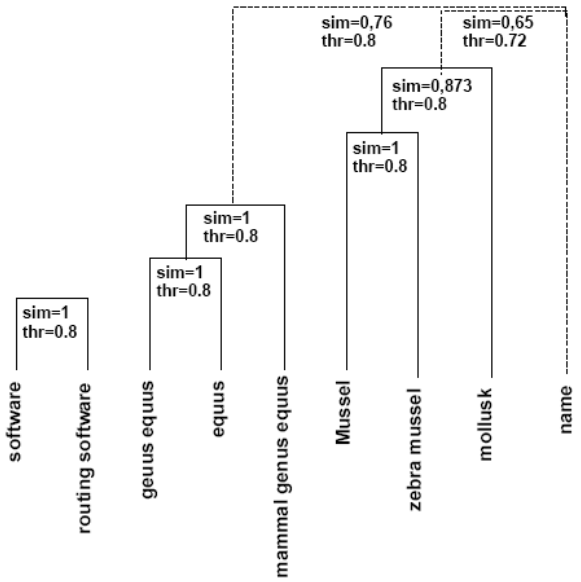

Figure 2. Group-average agglomerative hierarchical clustering of keyphrases. We show the pair of clusters merged at each iteration, with the corresponding similarity and threshold values computed from the similarity matrix in Table 1.

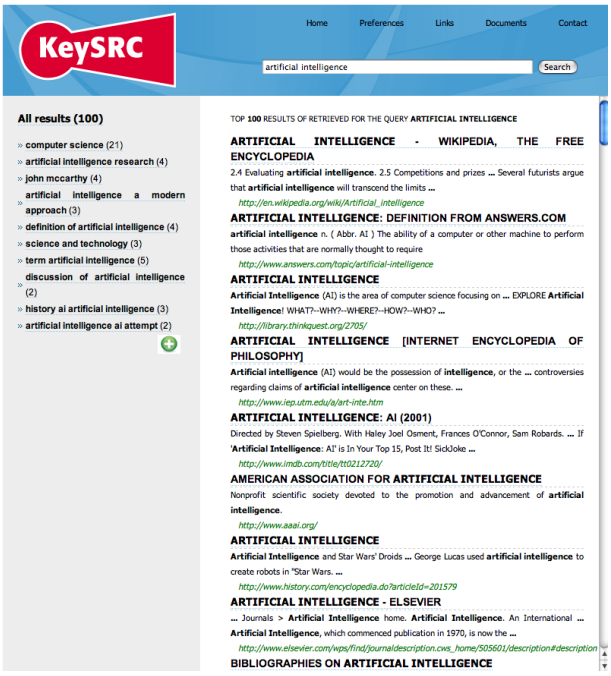

Figure 3. Results of KeySRC for the query “artificial intelligence”.

[8], spectral clustering [10], and graph theory [11]. Most closely related to our work are the systems based on GST. After Suffix Tree Clustering (STC) [7], 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 such representations for measuring document similarity while clustering ([20], [21]), or selecting the most informative GST nodes and then building the clusters around them ([22], [23]). 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 dendogram, 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 mostly interested in those which explicitly model the variance of subtopics in groups of documents, rather than trying to optimize the relevance of the set of documents to the whole query as in [24]. Most relevant to this paper is Essential Pages (EP) [4], 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., [3], [5]), 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, because it was used in our experiments.

The joint coverage score of a set of documents is expressed as a function of the coverage score of their constituent words, which, in turn, is related in a non-obvious manner to the relevance of the single word to the query. Words that have low or high relevance have little coverage, the remaining words are deemed important (the exact formula is given in [4]). Although EP has been devised for working with full documents, their authors suggest 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 pre-processing step.

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 result suggests that keyphrase-extraction algorithms based on a combination of n-gram filtration, lexical analysis and word frequency counts may be equally effective [26].

5. 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.

5.1. Tested systems

We tested the five following systems besides KeySRC.

- *STC*. Suffix Tree Clustering (STC) [7] has been described in the preceding section. In our experiments we used the version implemented in the open source Carrot[2] framework.
- *Lingo*. Lingo [9] is based on concepts extracted from search results via singular value decomposition.
- *CREDO*. CREDO [8] is based on the theory of concept lattices and is also available for PDAs and cell phones [18].
- *EP*. Essential Pages (EP) [4] has been described in the preceding section. Because we were interested in retrieving multiple documents per subtopic, in our implementation we not only selected the first ten essential pages (as with the basic EP) but we also recursively applied the algorithm to the rest of the search results.
- *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.

5.2. 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 re-ordering (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 subtopic relevance judgments. AMBIENT is fully described in [18] and is available at http://credo.fub.it/ambient for re-use by other researchers.

5.3. Results

The systems being tested were ran on the 100 search results associated with each AMBIENT query and the performance of the corresponding output was evaluated using the methodology described in Section 2. The results are shown in Table 2.

	kSSL (k=1)	kSSL (k=2)	kSSL (k=3)	kSSL (k=4)
KeySRC	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
EP	**13,79**	25,83	33,21	38,04
Yahoo!	14,18	31,58	40,78	48,12

Table 2. Performance of subtopic retrieval systems on the Ambient test collection expressed as kSSL for several values of k (best results in bold).

For $k = 1$, the best result was achieved by EP, followed by 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 Yahoo! is also 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 ranked list. 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 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 decrease in search length of the order of 30% for any value of k. The superiority of KeySRC over the other clustering systems was also clear, with dramatic improvements over STC and more limited but significant gains over CREDO and Lingo. KeySRC 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 and clearly better than Yahoo!. Recall, however, that we used a recursive variant of EP which favored retrieval of multiple documents per subtopic, whereas the results of the basic EP for $k > 1$ were much more similar to those of Yahoo!.

Table 2 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

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

Table 3. Relationships between cluster labels and subtopics.

results in Table 2 is that the increase of kSSL values seems to proportionally reduce as k grows. This phenomenon was due to the presence of some subtopics with very few 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 also the reason why we limited the maximum value of k considered in the experiment 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 3, 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.

6. 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.

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 achieved the best kSSL performance for retrieving multiple documents 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 $k = 1$).

Acknowledgment

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] B. J. Jansen, D. L. Booth, and A. Spink, "Determining the informational, navigational, and transactional intent of Web queries," *Information Processing and Management*, vol. 44, no. 3, pp. 1251–1266, 2008.

[2] 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*. ACM Press, 2003, pp. 10–17.

[3] 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*. ACM Press, 2006, pp. 504–511.

[4] A. Swaminathan, C. Mathew, and D. Kirovski, "Essential Pages," Microsoft Research, Tech. Rep. MSR-TR-2008-15, 2008.

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

[6] Y. Xu and H. Yin, "Novelty and topicality in interactive information retrieval," *Journal of the American Society for Information Science and Technology*, vol. 59, no. 2, pp. 201–215, 2008.

[7] 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*. ACM Press, 1998, pp. 46–54.

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

[9] S. Osiński and D. Weiss, "A Concept-Driven Algorithm for Clustering Search Results," *IEEE Intelligent Systems*, vol. 20, no. 3, pp. 48–54, 2005.

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

[11] E. Di Giacomo, W. Didimo, L. Grilli, and G. Liotta, "Graph Visualization Techniques for Web Clustering Engines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 2, pp. 294–304, 2007.

[12] W. Cooper, "Expected search length: A single measure of retrieval effectiveness based on weak ordering action of retrieval systems," *American Documentation*, vol. 19, no. 1, pp. 30–41, 1968.

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

[14] C. Carpineto, S. Osiński, G. Romano, and D. Weiss, "A survey of Web clustering engines," 2009, to appear in *ACM Computing Survey*.

[15] N. J. L. A. Andersson and K. Swanson, "Suffix trees on words," *Algorithmica*, vol. 23, pp. 102–115, 1999.

[16] E. Ukkonen, "On-Line Construction of Suffix Trees," *Algorithmica*, vol. 14, no. 3, pp. 249–260, 1995.

[17] 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.

[18] C. Carpineto, S. Mizzaro, G. Romano, and M. Snidero, "Mobile Information Retrieval with Search Results Clustering: Prototypes and Evaluations," *Journal of American Society for Information Science and Technology (JASIST)*, vol. 60, no. 5, pp. 877–895, 2009.

[19] M. A. Hearst, "Clustering versus faceted categories for information exploration," *Communications of the ACM*, vol. 49, no. 4, pp. 59–61, 2006.

[20] S.Branson and A. Greenberg, "Clustering Web search results using suffix tree methods," Stanford University, Tech. Rep. CS276A Final Project, 2002.

[21] H. Chim and X. Deng, "A new suffix tree similarity measure for document clustering," in *WWW '07: Proceedings of the 16th international conference on World Wide Web, Banff, Alberta, Canada*. ACM Press, 2007, pp. 121–130.

[22] 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*. IEEE, 2005, pp. 172–178.

[23] 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*. Springer, 2006, pp. 197–203.

[24] 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*. ACM Press, 2006, pp. 429–436.

[25] P. D. Turney, "Learning Algorithms for Keyphrase Extraction," *Information Retrieval*, vol. 2, no. 4, pp. 303–336, 2000.

[26] 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*. Springer, 2008, pp. 199–208.