Improving Web Clustering by Cluster Selection

Daniel Crabtree, Xiaoying Gao, Peter Andreae
School of Mathematics, Statistics and Computer Science
Victoria University of Wellington
New Zealand
daniel@danielcrabtree.com, xgao@mcs.vuw.ac.nz, pondy@mcs.vuw.ac.nz

Abstract

Web page clustering is a technology that puts semantically related web pages into groups and is useful for categorizing, organizing, and refining search results. When clustering using only textual information, Suffix Tree Clustering (STC) outperforms other clustering algorithms by making use of phrases and allowing clusters to overlap. One problem of STC and other similar algorithms is how to select a small set of clusters to display to the user from a very large set of generated clusters. The cluster selection method used in STC is flawed in that it does not handle overlapping clusters appropriately. This paper introduces a new cluster scoring function and a new cluster selection algorithm to overcome the problems with overlapping clusters, which are combined with STC to make a new clustering algorithm ESTC. This paper's experiments show that ESTC significantly outperforms STC and that even with less data ESTC performs similarly to a commercial clustering search engine.

Keywords: web clustering, cluster selection

1. Introduction

The amount of information on the Internet is growing at an enormous rate, and finding relevant information is becoming increasingly difficult. Current search engines allow a user to retrieve pages that match a search query, but the number of results returned by a search engine is often huge, and many of the results may be irrelevant to the user's goal. Search engines attempt to order the results to present pages that are more useful earlier, but the user will generally need to refine their search by adding to or changing the query to filter out the irrelevant results. The large ordered list of results provides little assistance to the user in this difficult query refinement process — the user may need to retrieve and scan many of the pages to determine the topics of irrelevant pages that need to be excluded by the refined query. A promising technique to address this problem is to organize the result set into clusters of semantically related pages so that the user can quickly overview the entire result set, and can use the clusters themselves to filter the results or refine the query.

Many clustering algorithms have been developed including K-means [5], Hierarchical Agglomerative clustering [1], and Suffix Tree Clustering (STC) [11]. When using only textual information (as opposed to also using additional information such as hyperlink information [8, 3]) for clustering, Zamir [10] has shown that STC outperforms other algorithms. The main advantages of STC over other clustering algorithms are that it uses phrases rather than words, and that it allows clusters to overlap.

Zamir [10] has found that using the full text of Web documents for clustering has better performance than using snippets returned by search engines, albeit at the cost of a large increase in processing time. However, using full text, STC returns far too many clusters. For example, using the first 300 documents (210 after removing pages without snippets, 404 errors, etc.) from a Google search on "jaguar", STC creates about 3000 clusters (compared to about 40 clusters using snippets), which is not very useful to the user since the user could not possibly handle so many clusters. One solution is to apply a score function to all clusters and only return the top clusters with high scores. However, neither the score function in [11], nor the slightly more advanced score function in [10] handle overlapping documents in clusters properly. These score functions work reasonably well when using only snippets, but when using full text the returned clusters are often dominated by closely related clusters that cover the largest topic. For example, among the top ten clusters for the "jaguar" data set, six of them are "cars" and the same documents appear many times, but many documents in other topics do not appear at all.

We addressed this problem by developing a new score function and a cluster selection algorithm to filter the clusters to maximize topic coverage and reduce overlap. Our new method, called ESTC (Extended Suffix Tree Clustering) significantly reduces the number of the clusters and heuristically determines and returns the clusters that should be the most useful to the user. This paper's experiments on clustering Google search results show that the new method ESTC outperforms STC and that ESTC is comparable to the commercial clustering search engine Grokker [2].

The rest of this paper is organized as follows: after describing the background (including a description of STC), the paper introduces the new score function and the cluster selection algorithm, then presents the experimental results, the conclusion, and directions for future work.

2. Background

This section first introduces the three stages of Suffix Tree Clustering (STC): document cleaning, building the suffix tree, and clustering. Then some of the recent improvements to STC are reviewed.

2.1. Three Stages

The document cleaning stage for most text-based document clustering algorithms is very similar. The HTML tags, punctuation and other similar non-informative text are removed; a set of stop words containing very common and uninformative words such as "the", "it" and "on" are removed; stemming is applied to reduce words to their root form, for example, "computer", "computing", "compute", "computers" are all changed to the root word "compute". In STC, the final step of this stage is to break down documents into phrases using standard sentence terminators such as full stops or exclamation points, and the location of surrounding HTML tags.

In the second stage, a suffix tree is created using the phrases in the documents (an example is shown in figure 1). The suffix tree is used to construct base clusters - sets of documents that contain at least one phrase in common. Each node in the suffix tree corresponds to a phrase, and each base cluster corresponds to a node on the suffix tree. For example, node A in figure 1 is the base cluster that represents documents containing the common phrase "the dog" and this base cluster contains documents 1 and 2.

In the third stage, a single-link clustering algorithm is used to combine the base clusters to form the output clusters. A link is added to connect two base clusters if they have sufficient documents in common — if, for both clusters, the fraction of documents in the cluster that are also in the other cluster is greater than a constant threshold (the similarity constant), then the two base clusters are connected by a link. The linked base clusters form a graph and a graph algorithm is run to find all the connected components. Each connected component consists of a set of base clusters whose union

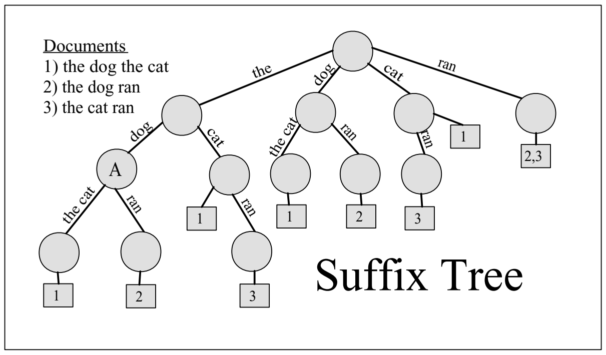


Figure 1. Suffix Tree

forms the output cluster. A scoring method is applied and the high scoring clusters are returned. More detail and discussion of the STC algorithm can be found in [10].

2.2. Recent Improvements

STC finds clusters based on the common phrases shared by documents. A suffix tree is a very efficient way to identify common phrases in documents, but suffix trees suffer from large memory requirements and poor locality characteristics. Suffix arrays [4], which have recently been applied to clustering, are similar to suffix trees and perform the same function, but have significantly lower memory requirements. Other improvements to aspects of the original STC algorithm have been made, for instance, SHOC (Semantic, Hierarchical, Online Clustering) [12] extends STC to handle oriental languages such as Chinese and introduces a mathematically well founded orthogonal clustering method that improves the STC method. Clustering performance can also be improved further by using other types of information such as hyperlink information [9].

3. ESTC: Extended Suffix Tree Clustering

The final step in the third stage of STC is to use a scoring function to select the clusters to output to the user. However, the current scoring function has problems and document overlap between clusters is not handled appropriately. This paper addresses these problems by developing a new score function and a new cluster selection algorithm. The cluster selection algorithm will be evaluated using STC, but could equally well be applied to the other clustering algorithms.

3.1. The current scoring function and its problem

Each base cluster is assigned a score based on the length of the common phrase and the number of documents in the base cluster. In [11], the score is defined as $s = |D| \cdot f(|P|)$ where $|D|$ is the number of documents in the base cluster, $|P|$ is the length of the common phrase, and $f(|P|)$ is defined as follows:

$$f(|P|) = \begin{cases} 0.5 & \text{if } |P| = 1 \\ |P| & \text{if } 1 < |P| \leq 5 \\ 6 & \text{if } |P| > 5 \end{cases}$$

Some implementations [10] weight this score by the sum of the term frequency multiplied by the inverse document frequency for each word; adding this extra weight is likely to be beneficial. For simplicity this weight is not used in this paper's testing, and the score is defined as in [11]. The score of a cluster is the sum of the scores of the base clusters contained in the cluster.

Our investigation shows that the score works fine for the base clusters but simply summing them up to form the score of a merged cluster is problematic — the score function over-counts the documents in overlapping base clusters, causing clusters formed by merging a large number of similar base clusters to be unreasonably favored. Since clusters are formed by merging base clusters with a high overlap, this over-counting is rampant. For example, if a cluster is formed from a large number of very similar base-clusters, then the score of the combined cluster will be very high, even though it is not significantly better than any one of the base clusters. This can lead to the selection of small clusters and clusters relating to a single dominant topic for output, while other less dominant topics are completely left out.

3.2. A new scoring function

To address the overlapping problem between base clusters, the score of each base cluster is split among the documents it contains: for a base cluster with score s containing $|D|$ documents, each document in the base cluster is allocated $\frac{s}{|D|}$, which is the same as $f(|P|)$ above. Then for a merged cluster, the score of a document is the average of the base scores it was allocated for the base clusters that form the merged cluster. The score of a merged cluster is the sum of the scores of its documents.

This new score function depends on the number of distinct documents in the cluster rather than the number of base clusters. Using this new score function to select the top n clusters provides better document coverage as demonstrated in the experiments in Section 4.

3.3. Cluster Selection Algorithm

The new score function can be used directly in selecting output clusters but the results are not fully satisfactory — the score function addresses the overlap between base clusters, but not the overlap between output clusters. The next step is to develop a cluster selection algorithm to search for the n clusters of most use to the user. Ideally the best n clusters should match the top n topics in the set of documents. However, the set of topics is external information that is not available to the clustering algorithm. The algorithm therefore uses the following heuristic to guide the search: the most useful set of clusters have minimal overlap and maximal coverage, so that the clusters are as distinct as possible (so users do not have to sift through the same documents multiple times) and most of the documents should be present (covered) in the clusters.

3.3.1. Heuristic The problem can be formulated as a heuristic search over the space of all sets of n or fewer clusters, with a heuristic function $H = D - \beta(C - D)$ where C is the sum of the sizes of the clusters, and D is the number of distinct documents. D represents the coverage of the solution, and $C - D$ is the number of overlapping documents in the solution. β is a constant used to balance the trade off between overlap and coverage. If $\beta = 0$, then there is no penalty for overlap and a cluster may be considered as long as it has distinct pages that are not already in the solution. For $\beta = 1$, a cluster is considered only if more than 50% of its documents are distinct. As $\beta \to \infty$, the percentage of documents that must be distinct approaches 100%, that is, no overlap is allowed. Based on our experiments, β in the range of $1 - 2$ has good performance and $\beta = 1$ is used in this paper's final evaluation.

3.3.2. Incremental Search and Look-ahead Ideally, the heuristic would be maximized by an exhaustive search that considered all combinations of n or fewer clusters. But an exhaustive search of all solutions is too expensive and since this heuristic only approximates the desired properties, an incremental search algorithm is used. The algorithm approximately maximizes the heuristic by starting with an empty set of clusters and extending that solution incrementally (adding one cluster in each step). In each step, a k-step look-ahead is used to select the cluster to be added to the current solution.

Simply selecting the best cluster (the one that adds the most to the heuristic value of the current solution, with ties broken by the scores of the clusters) at each step corresponds to a greedy search, which is equivalent to using 0-step look-ahead. However, the best cluster at the current step may not be the best selection in the long term because it may eliminate the possibility of adding other clusters that would make a better solution. In each step, to avoid making bad short term decisions, the algorithm looks ahead several steps before committing to a choice for the current step and adding the selected cluster to the current solution. The look-ahead is implemented using a depth limited, branch and bound search from the current solution. The depth limit specifies the number of steps of look-ahead. A k-step look-ahead considers all possible 1, 2, ..., $k+1$ cluster extensions to the current solution, and the best cluster of the best extension is chosen to be added to the current solution.

The cluster selection algorithm was tested using different amounts of look-ahead and the results were compared. The results in figure 2 (expressed in terms of F-measure defined in section 4.1) show that some look-ahead is clearly worthwhile, but there is no advantage in using larger amounts of look-ahead. Note that there is a large time cost when using three step or greater look-ahead — 3-step look-ahead increased computation time to over an hour, in contrast to less than a second in all cases with less look-ahead. The final experiments use a two step look-ahead.

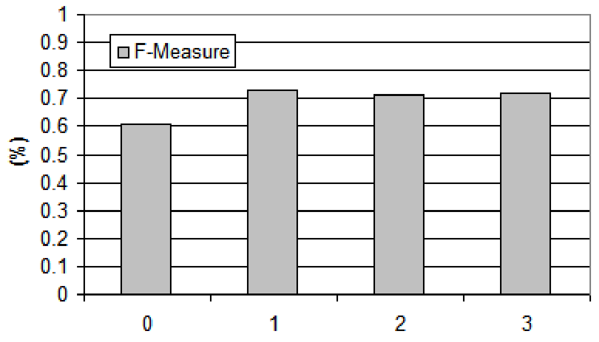


Figure 2. Performance with varying steps of look-ahead

3.3.3. Pruning A number of techniques are used to further improve the efficiency of the cluster selection algorithm. In order to reduce the search space, the clusters are sorted by their score and the clusters whose score is too low are removed. The cut-off threshold depends on the total number of clusters, the cluster scores, and the number of clusters expected in the solution.

A more sophisticated pruning technique that incorporates branch and bound pruning is used to reduce the search space further. Branch and bound pruning is implemented by pruning any branch that cannot possibly influence the final solution; determined by comparing the best known extension (the one that adds the most to H) against the maximum possible extension from the current branch. The maximum possible extension is an upper bound on the extension from the current branch and is computed under the assumption that there is no overlap between the partially constructed extension and future extensions.

After each step, the clusters are re-ordered first by the increase in H from adding the cluster to the current solution, then secondarily by cluster score. In the search algorithm, the most costly atomic operation is computing the intersection between a cluster and the current extension to the current solution. Due to the constraints imposed by the ordering, many branches can be pruned without computing the intersection, some branches can be pruned without any further consideration, and permutations can be automatically eliminated.

3.3.4. Cluster Selection Example Figure 3 shows a set of 4 clusters: A, B, C, and D. A, B, and C are disjoint and D overlaps with them. The numbers show the number of pages in each region and clusters A, B, C, and D have 35, 34, 33, and 60 pages respectively. In the following examples, the cluster selection algorithm uses $\beta = 1$, which means a cluster is considered for addition to the current solution only if more than 50% of its documents are distinct.

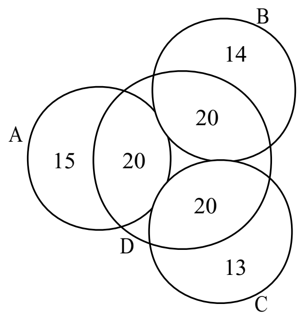


Figure 3. Cluster Selection Example

In the case where there is no look-ahead, the search starts with an empty current solution {}. When picking the cluster for the first extension, there is no need to even consider A, B, or C, since the clusters are ordered as D, A, B, C, and the first cluster D is the best. Once D is added to the current solution, the search ends and the final solution {D} is just a single cluster. No further step is taken to add any other clusters, as all clusters decrease H when added to current solution {D}. However {D} is not the optimal solution: {A, B, C} is superior because it has more coverage. {A, B, C} can only be found by using look-ahead.

With 1-step look-ahead, in order to select the first cluster, all possible combinations of 1 or 2 clusters are considered as extensions to the current solution {}. The extensions considered are: {D}, {D, A}, {D, B}, {D, C}, {A}, {A, B}, {A, C}, {B}, {B, C}, and {C}. The best extension is {A, B}, therefore cluster A (because it is better than B) is selected as the first cluster to be added to the current solution. Similarly in the second step, the extensions {B}, {B, C}, {B, D}, {C}, {C, D}, and {D} are considered and cluster B is selected from the best extension {B, C} to be added to the current solution, giving a current solution of {A, B}. In the third step, the extensions {C}, {C, D}, and {D} are considered and cluster C is selected from the best extension {C} to be added to the current solution. As D is the only cluster left and adding D to the current solution would decrease H, the algorithm terminates and the final solution is {A, B, C}.

Pruning has made the algorithm more efficient and due to pruning much fewer extensions are considered. In step 1, the clusters are ordered as D, A, B, C and the extensions considered are {D}, {D, A}, {D, B}, {D, C}, {A}, and {A, B}. Once {A, B} is identified as a better extension to the current solution than {D}, there is no need to consider {A, C} since there is no way this could exceed {A, B} due to the ordering and the lack of overlap between A and B (if there were any overlap between A and B, {A, C} would be considered). For similar reasons, {B} and {C} do not need to be considered as there is no way they could outperform A. A is then selected and added to the current solution {}. If 2 step look-ahead were used, there would be even more pruning. For example, although {D, A} is considered, it is worse than D, so the branch is pruned and {D, A, B} and {D, A, C} would not even be considered.

In step 2, the current solution is {A} and the remaining clusters are reordered as B, C, D: due to the overlap between D and the current solution {A}, D adds at most 20 to H, whereas B and C add at most 34 and 33 respectively. Once the extensions {B} and {B, C} have been considered, there is no need to consider any other extensions since these cannot outperform {B, C}. B is then selected and added to the current solution. In step 3, D is eliminated during the ordering process because of its overlap, so C is added to the current solution {A, B} without considering {D} explicitly. The final clustering and solution is thus {A, B, C} and only 9 extensions are considered, compared to 19 without pruning.

4. Experiments

4.1. Evaluation method

We evaluated ESTC by comparing its performance against three other algorithms. The performance was measured using a gold-standard approach [6] in which the clustering from the algorithm is compared to a manually constructed ideal clustering in which each ideal cluster corresponds to a topic. The topics in the ideal clustering should correspond to the highest level topics in a topic hierarchy. For example, in a search for 'jaguar', topics may include 'car clubs', 'car parts', 'car dealers', and 'animals'; the highest-level topics assigned for these should be 'car' and 'animal'.

The evaluation measure was based on precision and recall [7] – two standard evaluations methods used in information retrieval and information extraction. Precision measures how accurately the clusters from the algorithm represent the topics in the ideal clustering, and recall measures how many documents in the ideal clustering are covered by the clusters from the algorithm. The precise formulation of the measures is given below.

Each cluster from the algorithm is labeled with the topic of which it has the largest number of documents. Let C be the set of clusters from the algorithm, and T be the set of topics from the ideal clustering. $C_t \subset C$ is the set of clusters that are labeled with a particular topic $t \in T$.

The precision $P(c)$ of a cluster $c \in C$ is given by

$$P(c) = \frac{\max_{t \in T}\{|D_{c,t}|\}}{|D_c|}$$

and the recall $R(t)$ of a topic $t \in T$ is given by

$$R(t) = \frac{|\bigcup_{c \in C_t} D_{c,t}|}{|D_t|}$$

where D_c is the set of pages in cluster c, D_t is the set of pages in topic t, and $D_{c,t}$ is the set of pages in cluster c that are labeled with topic t.

The overall precision P and recall R of a clustering are the average of the individual cluster precisions and topic recalls weighted by the size of each cluster and each topic respectively:

$$P = \frac{\sum_{c \in C} P(c)|D_c|}{\sum_{c \in C} |D_c|}$$

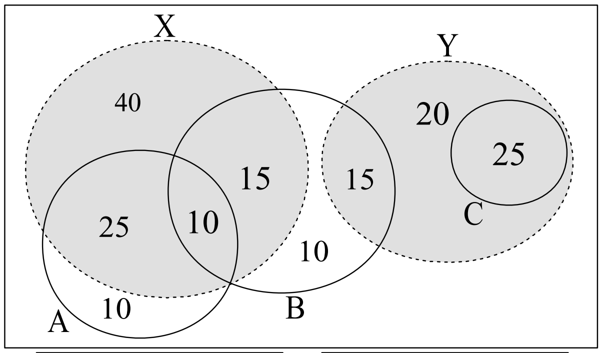
$$R = \frac{\sum_{t \in T} R(t)|D_t|}{\sum_{t \in T} |D_t|}$$

The standard F-measure F is used as the overall measure of how well the clustering matches the ideal clustering:

$$F = \frac{2PR}{P + R}$$

The Google search results used for evaluation often contain some irrelevant documents which are noise and really should not be in the search results. These documents typically form several topics with very small sizes — less than 3. For our evaluation measure, very small topics that contain less than 4% of all documents are excluded from computation of the overall recall R.

An example is given in figure 4. X and Y are topics, A, B, and C are clusters, and the numbers specify the number of documents in each region (the documents outside topics X and Y are in very small topics that are excluded from computation of overall recall R). It is clear that clusters A and B represent topic X and cluster C represents topic Y. The precision of the clusters and the recall of the topics can be computed as shown in figure 4 and from these it is found that overall precision is 70.83%, overall recall is 50.00%, and F-measure is 58.62%.



Cluster	Precision
A	35/45
B	25/50
C	25/25

Topic	Recall
X	50/90
Y	25/60

Figure 4. Evaluation Example

4.2. Comparison between ESTC and STC

ESTC was tested by clustering Google search results. Two test data sets were created: the search results for “jaguar” and the search results for “salsa”. All the documents were retrieved and each document was manually assigned to a topic. The salsa documents were quite difficult to classify, which increases the margin of error in the salsa results and this should be taken into consideration when examining the results. The jaguar data set consists of 210 documents assigned to 34 topics, and the salsa set consists of 198 documents assigned to 21 topics (by chance, in both data sets, 171 documents and 4 topics were significant to recall evaluation). The data sets include the full text of the documents, not just snippets. The data sets are available online at http://www.danielcrabtree.com/research/wi05/rawdata.zip.

Three algorithms were compared: STC, STC-NS, and ESTC. ESTC uses both the new score function and the cluster selection algorithm to select output clusters. STC-NS is an intermediate algorithm that uses the new score function but not the new cluster selection algorithm. The top 10 clusters returned by the three algorithms were evaluated with eight different values of the similarity constant. The results are shown in figures 5 and 6.

The results on both data sets show that ESTC clearly outperforms STC and STC-NS in all the similarity range except two points in the salsa case. In the 2 cases where STC outperformed ESTC, the recall for ESTC was higher and precision lower. A detailed analysis of the precision and recall

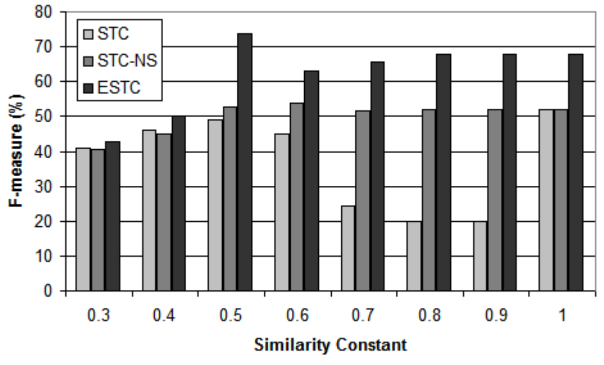


Figure 5. Jaguar Results

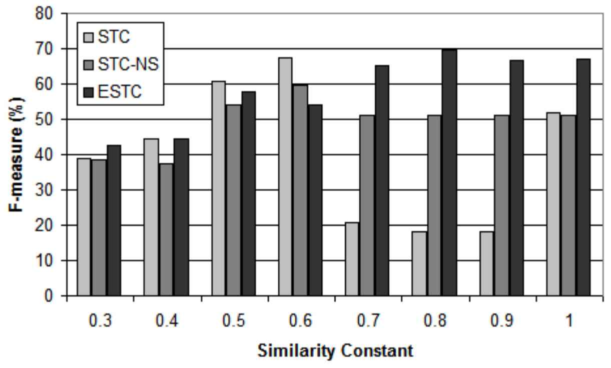


Figure 6. Salsa Results

across all results shows that ESTC significantly improves the recall with a slight drop in precision. The best performance in the jaguar case is achieved by ESTC with a similarity constant of 0.5 and the best performance in the salsa case is achieved by ESTC with a similarity constant of 0.8. It can also be seen that STC-NS works better than STC most of the time.

ESTC and STC-NS are both far less sensitive to the similarity constant than STC. The similarity constant is based on the overlap between base clusters and it controls the probability of base clusters being merged; altering the similarity constant affects the overlap between base clusters in the merged clusters. The significantly reduced sensitivity to the similarity constant in ESTC therefore confirms our earlier hypothesis that there is a problem with the treatment of overlap in STC and that ESTC solves the problem.

4.3. Comparison between ESTC and Grokker

Grokker [2] is a commercial visual information management framework. One of the components of the current system is a meta search engine with clustering available at http://www.groxis.com. The search results are presented as a set of clusters. When the user clicks on one cluster, the cluster is expanded and the documents or sub-clusters are displayed. The clustering component of Grokker is a hybrid of k-means and an agglomerative clustering method. The clustering algorithm identifies and uses vectors of noun-phrases from the page titles and search engine snippets. Grokker finds hierarchical results by recursively applying the algorithm to the clusters found at a given level.

To compare Grokker with ESTC, a search for "jaguar" was performed on Grokker with it set up to obtain results solely from Google. The clusters are organized hierarchically and each level contains a "more categories" cluster which consists of documents that are not clustered at the current level. These hierarchies were flattened into their top level clusters with any duplicated documents within any single cluster removed and the top-level "more categories" cluster removed. The resulting clustering consisted of 9 clusters. Comparing the clusters with the manually assigned topics, Grokker achieves 77.5% precision, 51.5% recall, and 61.9% F-measure [1]. ESTC was also tested on Google "jaguar" search results, which is very similar to the data set used in figure 5 except that snippets rather than full text are used for clustering. The top 10 clusters of ESTC achieve 80.0% precision, 46.0% recall, and 58.4% F-measure which is comparable to that of Grokker.

Note that Grokker uses both snippets and page titles for clustering but ESTC in this test only had access to the snippets. Although the ESTC method was using less data in this test (which explains its higher precision and lower recall), ESTC achieved a comparable F-measure. Note that ESTC's performance on full text achieves a maximum 74% F-measure as shown in figure 5, which is much better than that of Grokker.

5. Conclusions

The paper has described ESTC — an improvement to the Suffix Tree Clustering algorithm that uses a new score function and a new cluster selection algorithm that maximizes topic coverage and minimizes cluster overlap. The experiments showed that the clustering performance of ESTC is significantly improved over STC. ESTC was also compared with a commercial clustering search engine showing that even when using less information ESTC achieves comparable results.

One possible direction for future work is to further improve the cluster selection method by investigating heuristic functions that take account of other information in addition to cluster overlap and document coverage. One idea for the heuristic function to incorporate cluster quality information directly, rather than indirectly through the scoring method. There are also many other directions for improvement in the STC algorithm. For example, developing a new algorithm for the third stage of STC in which the base clusters are merged.

1 The test was run several times and the search results of Grokker changes slightly each test run. Another test returned 10 clusters and achieved 76.0% precision, 56.0% recall, and 64.5% F-measure.

Acknowledgements

Daniel Crabtree is supported by a Top Achiever Doctoral Scholarship from the Tertiary Education Commission of New Zealand.

References

[1] R. Ali, U. Ghani, and A. Saeed. Data clustering and its applications. `http://members.tripod.com/asim_saeed/paper.htm`.

[2] Groxis. Grokker - http://www.groxis.com/service/grok, 2004.

[3] J. Hou and Y. Zhang. Utilizing hyperlink transitivity to improve web page clustering. In *Proceedings of the Fourteenth Australasian database conference on Database technologies, Adelaide, Australia*, volume 17, pages 49–57, 2003.

[4] K. Malde, E. Coward, and I. Jonassen. Fast sequence clustering using a suffix array algorithm. In *BIOINFORMATICS*, volume 19 (10), pages 1221–1226, 2003.

[5] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[6] P. Tonella, F. Ricca, E. Pianta, C. Girardi, ITC-irst, G. D. Lucca, A. R. Fasolino, P. Tramontana, U. di Napoli Federico II, Napoli, and Italy. Evaluation methods for web application clustering. In *5th International Workshop on Web Site Evolution, Amsterdam, The Netherlands*, September 2003.

[7] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[8] Y. Wang and M. Kitsuregawa. On combining link and contents information for web page clustering. In *13th International Conference on Database and Expert Systems Applications DEXA2002, Aix-en-Provence, France*, pages 902–913, September 2002.

[9] Y. Wang and M. Kitsuregawa. Use link-based clustering to improve web search results. In *Proceeding of 2nd International conference on Web Information System Engineering (WISE2001), IEEE Computer Society*, pages 115–124, December 2002.

[10] O. Zamir. *Clustering Web Documents: A Phrase-Based Method for Grouping Search Engine Results*. PhD thesis, University of Washington, 1999.

[11] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.

[12] D. Zhang and Y. Dong. Semantic, hierarchical, online clustering of web search results. In *Proceedings of the 6th Asia Pacific Web Conference (APWEB), Hangzhou, China*, April 2004.