TECHNICAL REPORT

Report No. CSAI2005-01
Date: March 2005

Automatic Document Clustering using Topic Analysis

Robert Muscat

Department of Computer Science & A.I.
University of Malta
Msida MSD 06
MALTA

Tel: +356-2340 2519
Fax: +356-2132 0539
`reports@cs.um.edu.mt`
`http://www.cs.um.edu.mt/~reports`

Automatic Document Clustering using Topic Analysis

Robert Muscat
University of Malta, Malta.
rmus018@um.edu.mt

Abstract: *In this work we look at the automated organisation of documents in a document base into clusters and cluster hierarchies. We apply topic segmentation to detect topics within documents and using term relationships attempt to build hierarchies which represent a "real world" topic hierarchy. Finally, we assign documents to each of these topics using a standard clustering technique.*
We also propose two evaluation methods for document clustering systems. The first is an adaptation of tree measure algorithms to document hierarchies. The use of this method will require a pre-defined tree which has been agreed upon as a suitable benchmark. The second is independent of any benchmark trees and presents the evaluator with a number of measures which allow him to assess the properties of the tree.

Automatic Document Clustering using Topic Analysis

Robert Muscat
University of Malta, Malta.
rmus018@um.edu.mt

Abstract: *In this work we look at the automated organisation of documents in a document base into clusters and cluster hierarchies. We apply topic segmentation to detect topics within documents and using term relationships attempt to build hierarchies which represent a "real world" topic hierarchy. Finally, we assign documents to each of these topics using a standard clustering technique.*
We also propose two evaluation methods for document clustering systems. The first is an adaptation of tree measure algorithms to document hierarchies. The use of this method will require a pre-defined tree which has been agreed upon as a suitable benchmark. The second is independent of any benchmark trees and presents the evaluator with a number of measures which allow him to assess the properties of the tree.

1 Introduction

As the organisation of the web keeps revolving around search engines, the typical lists provided by these services is too vast to be handled by a single human being. Document clustering is promising itself as a handy tool to browse this information without the user suffering from cognitive overload. We discuss an approach to automatically extract topics from documents and avoids the need of indexers to structure documents in hierarchical clusters or to prepare training sets for typical machine learning algorithms. Indexers cannot guarantee consistent agreement between them and as an example Uren reports a typical agreement rate of 41$\pm$2% between indexers indexing the same data [Ure00]. This can perhaps give enough good reasons to try to automate such tools in order to remove the non-deterministic human element.

During our work, research led us to the fact that evaluating such tools is still a vague issue. We try to offer new means of evaluating these approaches by borrowing from other areas (like tree distance algorithms [Sel77, Tai79, Kle98, ZS89, KTSK00, Bil03]) and extending on existing work (namely [SL00]).

2 Review

In this section we will give a very brief overview of previous related work. We look at two the main aspects tackled in our work separately, *Document Clustering* and *Evaluation Techniques.*

2.1 Document Clustering

Scatter/Gather was one of the first attempts to group documents into clusters. The interaction was quite simple. Initially the user is presented with a set of groups which have been determined by the *Fractionation* algorithm. The user selects groups from the list presented, these groups are then merged and reclustered using another algorithm, *Buckshot* (which is faster but less accurate). The idea is that by iterating this process and selecting different clusters each time the user is revealed different views of the domain [CKPT92, HP96, CKP93]. Both of the two algorithms used are an adapted version of *k-means.* Further work on scatter/gather has improved the speed at which the user navigates through hierarchies (by using the *Cluster Refinement Hypothesis* which essentially states that most of the elements in two similar clusters will end up in the same cluster but higher up in the hierarchy).

Another important approach was that taken by Zamir and Etzioni [ZE98, ZE99]. Their algorithm *Suffix Tree Clustering* lies between an information retrieval system (search engine) and the user, and linearly analyses the results of the query to provide a structure to the stream of data. The algorithm divides the clustering process into three stages, *preprocessing*, *base cluster identification* and the *combine base clusters* stage before it actually provides the user with a list of clusters. The main advantage of the algorithm is that it starts generating data as soon as the stream is being processed, so this means that practically the user can be provided with the data as soon as he makes a request. As the name implies, the *preprocessing* stage processes the stream for analysis by removing tags, punctuation and transforming terms into stems. The second phase, *base cluster identification* builds a suffix tree out of the stems from the first phase. Each of these nodes is assigned a score which increases with the number of documents which hit the nodes and decreases linearly with the length of the phrase in the node. The final phase, *combine base clusters* analyses the nodes to join similar ones with edges and spares the user from seeing duplicate data. Finally, the set of available graphs are scored and the top ten are returned to the user. Grouper was an online application of the *suffix tree algorithm* and used results from *HuskySearch*, a meta-search engine, which was under development by the same team.

Other work by Lerman in [Ler99] and Schütze *et al.* in [SS97], takes a more standard approach to clustering but concentrate their work on the preprocessing of the document space. They concentrate on reducing the dimensions of documents in order to make clustering faster without deteriorating quality. Schütze tests two approaches, document truncation (taking the top 20 and 50 terms) and Latent Semantic Indexing (LSI is a method to project documents in a smaller dimension space while retaining the maximum information possible). Their results show only a difference in time efficiency which was not expected, since LSI[DFL$^+$88, DDL$^+$90] has generally improved performance of the information retrieval system. Lerman [Ler99] shows that LSI does in fact improve cluster quality. Her work shows that in order for this to happen, the dimension size selected must be optimal (or near optimal). Lerman suggests a method to select an ideal dimension by selecting the point in the vector space where the largest weight gap occurs. Her work shows improvement on cluster quality (rather than time efficiency) with LSI using a simple single link algorithm.

A number of other approaches exist which we will not tackle due to space restriction. We suggest the reader the following [BM98, jKgL02, Lan95, LGXZ02, MS00, PL02, ST00, WC00] for further reference.

2.2 Evaluation

Evaluating document clustering is still a grey area with no accepted technique available, especially for cases where the hierarchy is unknown and has to be generated.

The most common evaluation applied is using standard information retrieval values like *recall* (percentage of relevant retrieved documents over total retrieved) and *precision* (percentage of relevant retrieved over total retrieved). This is ideal for techniques where the hierarchical structure is known and accepted before hand, and we can compare the entries for each cluster from the test against the correct data. To be able to calculate recall and precision clusters have to be in some way labelled.

The issue is more complex when the overall hierarchy is not available. A number of attempts have been made to define measures which objectively allow clustering algorithms to be compared but there still is no accepted standard. In most cases a function is defined which determines the quality of a cluster but this is not objective and is only an independent measure which is not gauged against any other reference except itself.

A number of datasets have emerged out of which a few are growing to become accepted standards, but still these do not take in consideration algorithms which generate the hierarchy themselves. Currently the `Reuters-21578`[1] data is the most widely used. Work is underway to add another reuters data set which is the `Reuters Corpus`[2] which is a larger version and officially released by Reuters. The `Reuters Corpus` is being pushed to replace the `Reuters-21578` data set.

The closest method which is relevant to our requirements is that used by Mandhani et al. in [MJK03]. The authors combine two methodologies to evaluate their work, the first considers each cluster as a single entity and a measure is used to analyse the quality of its content (the two suggested are *entropy* and *purity*). Secondly, they analyse the resulting tree, hierarchically at each level, by looking at the number of nodes per level, the purity at each level and by comparing the generated node labels at each level. We think that this kind of hybrid analysis is the best available approach which can be applied to automatic document clustering. This approach though generates a large number of results (separate values per level). An obvious enhancement would integrate all these separate results in fewer (ideally one) values. In our work these labels are not available and hence we cannot guarantee that this mapping is always correct. This forced us to approach the evaluation process with different issues in mind.

3 Our Work

3.1 Automatic Clustering

Consider the case where we are given a set of documents and we only have access to the contents of the data i.e. we can actually read the documents. We now want to be able to group each of these documents with other documents on the basis of their topic. In this case we are doing what a human indexer does. Except that professional indexers have behind them years of experience, formal education and specific standard techniques to approach this particular kind of problem. Let us now consider the exact same scenario only that this time the indexer is totally unknowledgable about the topics of discussion in the topic set. We can safely presume that if the indexer reads the document set and analyses the contents of each document by looking at term co-occurrence and other common features, she can come up with some form of topic hierarchy and document clusters which, although not necessarily the ideal solution, would presumably satisfy a number of judges [Ure00]. This also follows in consideration of the traditional information retrieval notion that if a document contains the same terms found in a query, then that document is relevant to the

[1]http://www.daviddlewis.com/resources/testcollections/reuters21578/
[2]http://about.reuters.com/researchandstandards/corpus/

query [Rij99].

We assume that one document, in general, spans several topics of discussion, even if not all of them are discussed in detail. To gather enough information which makes building a tentative hierarchy a well informed task, we suggest an approach similar to passage-level retrieval. In [Cal94] a document is subdivided into smaller divisions, either paragraphs or windows of specific sizes. Term weightings are also re-calculated to enhance the significance of a particular terms in the context of one document. [Cal94] tests with both approaches, paragraphs and windowing.

Specific terms in each of these subdivisions must be identified using some particular technique e.g. taking the top three terms with highest `TF`$\times$`IDF`. What is required is for the chosen terms to be highly representative of topics referred to within the document. After we have specifically identified topic terms within these passages, the next phase is to find parent, child and sibling relationships between each topic. A top level parent (root node) is required to act as a common parent to all topics which have absolutely no pertinence to each other.

Finally, the documents are traversed and assigned to the tree node which is most relevant to the general contents of the document. One can also consider assigning a relevance score which directly takes in consideration the topic contents of the document against the node being considered. Documents similar (e.g. above a threshold distance measure) to the nodes in the tree can be associated to that cluster (a node is the representative of a potential cluster). This latter approach also allows documents to be assigned to a number of clusters, rather than uniquely assigned to one cluster.

In the next sections we will look in more detail at the reasoning behind these separate sections while we pinpoint the details for our implementation.

3.1.1 Document Segmentation

Documents tend to mention a number of topics within one discussion. Looking at a document as one "atomic" piece of data can only suggest the general topic of discussion. These topics of discussion are also not necessarily related in a real world scenario, consider for example a document talking about physics which uses examples of two trains leaving two cities. Although they are not related, the document is useful to learn about the existence of physics, trains and railways. The longer a document the higher the probability of it spanning a larger number of topics.

By looking at segments of one document, we can choose terms which although not necessarily important globally, (e.g. the term does not have a high `TF` or `TF`$\times$`IDF`) they are enough to indicate the existence of topics. We use TextTiling as our main topic segmentation algorithm [Hea, Hea94].

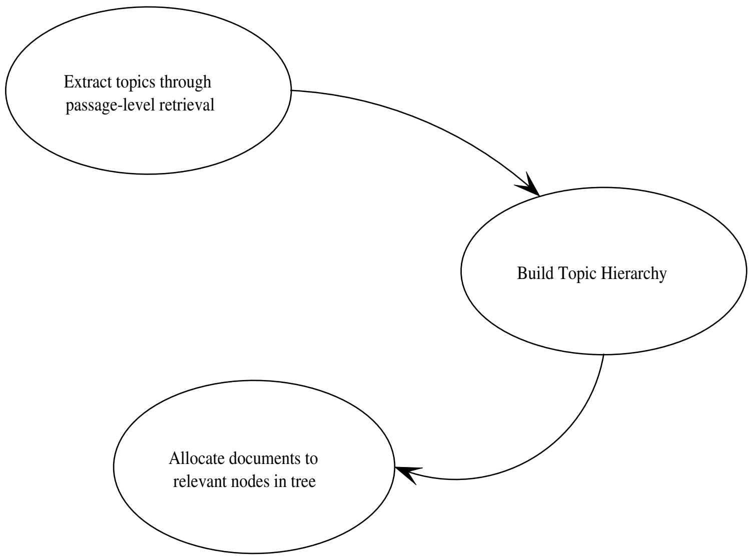
Extract topics through passage-level retrieval
Build Topic Hierarchy
Allocate documents to relevant nodes in tree

Figure 1: Framework
Suggested Approach

3.1.2 Identifying Topic Terms

Identified document segments need to be analysed so as to find topic terms. Such candidate topic terms have special features within the document set. Using terms with high $\texttt{TF}\times\texttt{IDF}$ will select only those terms which are good discriminators across the document set. These terms are not ideal candidate terms. We do not want terms which discriminate one particular document, but rather terms which discriminate topics. We identify the main and ideal properties of these terms,

- Consistent co-occurrence with other terms
- Some of them occur in most segments within one (or some) document and occur rarely in others

After a number of tests we specify in [Mus05] we opted for the $\texttt{TF}^2\times\texttt{IDF}$ value.

3.1.3 Determining a Hierarchy

Having a list of candidate topic terms at hand is the basic important step. Assuming all the topic terms have been extracted from the domain and are available in this list, statistical relationships can be extracted from the use of these terms in the domain to determine relationships across them.

A number of techniques exist to extract this kind of information such as using particular "key" words, using co-occurrence or pre-specified structures either built automatically (using machine learning techniques) or manually. Obviously such manual structures suffer from fallacies examples of which have been outlined in [Mus05].

These relationships can be used to build hierarchical tree structures which represent how topic terms interrelate, starting from the most general (as a root) to the most specific (as leaves).

We utilise `subsumption` [SC99] as our main algorithm to find relationship between terms and from there construct a term/topic hierarchy.

3.1.4 Clustering

The clustering process principally involves assigning documents to nodes in the tree, and always giving preference to the most specific node (deepest) which is relevant to the document. Clustering algorithms are available in many areas of research from imaging technologies to data warehousing. The type of clustering algorithms we are interested in are those which require a predefined knowledge (or assumption) of the partitions, which in our case are the actual nodes in the tree. These algorithms use this knowledge as a tentative partitioning of the domain. This partitioning changes while documents are being assigned.

In our work, the tree is passed on in a hierarchical structure which can be viewed linearly as a list of queries. We use Ward's [Jr.63] to assign documents to clusters in conjunction with the *Cosine Similarity Measure*. We also apply Rocchio's term reformulation (cited by [BYRN99]) to this process to refine the quality of the queries.

3.1.5 Cluster Naming

Finally, we should bear in mind that the ultimate goal of such a system is to make the searching user's life easier. Hence, the presentation of document centroids as cluster representatives is not exactly an ideal approach. It tends to be confusing, not self-explanatory and overwhelming. This will definitely scare off people from utilising the framework.

An area in information retrieval today studies ways of summarising text by taking sentences and paragraphs which the algorithm determines to be representative of the whole text. By assuming that each cluster is a document domain unto itself, such algorithms can be applied to this domain and return summarisations of it, to present to the user.

We apply these ideas by extracting important phrases from our cluster (in our case these could be sentences or four word phrases which have centroid terms) and then scoring each of these phrases with respect to the whole cluster and collection. The top ranking phrase/sentence is used as a cluster label (similar to [Wal03]).

3.2 Evaluation

From our review in [Mus05] we conclude that there is no generally accepted way to evaluate document clustering techniques. Available test sets which are considered as a standard are primarily geared to approaches where more than a test set, a learning set is required. In our work we propose two evaluation methods which work around the major issues which we consider problematic in these sets.

Our first approach took in consideration the generated hierarchy and a hierarchy which has been agreed as a good one. This is common to the general idea behind relevance judgements which are test sets which offer "queries" and related documents, ideal for testing. In our work, we suggest the application of tree distance algorithms to estimate the edit distance between the two trees and hence provide a measure of how much the two trees differ. The edit distance is the amount of insertions, deletions and replacements required to convert one tree into another. We propose two tests using this approach. The first is a comparison of both hierarchies as a whole. The second is a multilayered comparison where iteratively the two trees are compared and collapsed from the leaves upwards. We propose that documents are viewed as tree nodes and hence this comparison will take in consideration all the aspects of the tree which is determined by both the nodes and the placement of the documents to tree nodes.

Unfortunately, due to the nature of available edit distance algorithms comparing two large trees results in intractable times. We have therefore worked on the assumptions taken in [SL00] where trees are evaluated by extracting measures which are directly related to the browsing user. In [SL00] the "traversal algorithm" extracts the depth of a document in the tree. We also use the following measures divided in two categories, tree measures and cluster quality measures. The tree measures are,

Maximum Depth The maximum depth of the tree. This measures the worst case path the user has to take to navigate to his required document as in [SL00].

Maximum Node Breadth The maximum number of siblings which are not documents in the tree. This represents the worst case choice size given to the user in order to proceed with his search (if it is not yet completed).

Maximum Document Breadth The maximum number of sibling documents in the tree, represents the maximum number of documents to choose from a topic/node.

Average Depth The average depth across all branches represents the overall average path length to a document.

Average Node Breadth The average number of siblings which are not documents in the tree. This implies the overall average number of decisions to be taken to move on in the search at each level.

Average Document Breadth The average number of sibling documents in the tree representing the average number of documents the user has to face per node.

Tree Measure is a value which represents the ratio between depth and breath of the tree, and essentially is the ratio of documents per level. We calculate the measure using

$$\tau = \frac{\text{AverageDocBreadth}}{\text{AverageDepth}}$$

A ratio $1 : \tau$ specifies a normalised average of nodes per level. This can extended by also taking in consideration nodes.

Cluster measures,

Number of documents (`SIZE`) The number of documents in cluster C i.e.

$$\alpha = |\mathbf{C}|$$

Maximum Distance From Centroid (`MDFC`) This value is the maximum distance of any element in the cluster from the centroid. Thus all other documents lie within this distance.

$$\delta = \max_{i \in C}(\text{dist}(C, d_i))$$

Average Distance From Centroid (`ADFC`) The distance of all documents from the centroid is averaged (overall similarity). This was shown by Steinbach [SKK00] to be the same measure as the average pairwise distance between all documents (β and γ).

$$\beta = \frac{1}{|\mathbf{C}|} \sum_{d \in \mathbf{C}} \text{dist}(d, \mathbf{C})$$

$$\gamma = \frac{1}{|\mathbf{C}|^2} \sum_{i,j \in \mathbf{C}} \text{dist}(d_i, d_j) = \|c\|^2$$

The measure is used here to analyse the characteristics of the clusters. The two internal measures as specified by [ZK02] are identical to β and γ but adapted to include all available clusters with each value normalised with respect to their size. These adaptations therefore analyse the result as a whole (rather than individual clusters). After applying these adaptations, the function of these measures is no longer identical.

Density (`DENS`) measures the concentration of documents in a cluster. Clusters with many documents close together have a high density. Sparse clusters (in relation to a large area) have a low density.

$$\epsilon = \frac{|\mathbf{C}|}{\max_{i \in C}(\text{dist}(\mathbf{C}, d_i)) - \min_{i \in C}(\text{dist}(\mathbf{C}, d_i))}$$

4 Main Results

In this section we will outline our **main** results.

4.1 Interpreting the Results

We use the `Reuters-21578` data set as our document corpus. Three data sets have been created from `Reuters-21578`. We will be referring to them as `Reuters-A`, `Reuters-B` and `Reuters-C` (Table 1). In our tests each of the multiple files in each set creates one tree and one set of results. We average our values as a global across the particular set.

Data Set	Description
`Reuters-A`	Original division, 21 files of 1000 articles and 1 of 578.
`Reuters-B`	Created 10 files of 2000 articles and 1 of 1578.
`Reuters-C`	Created 5 files of 4000 articles and 1 of 1578.

Table 1: Data sets derived from `Reuters-21578`

We refer to our parameter sets as outlined in Table 2.

Parameter Set	Description
ParA	3 Term Per Segment, Choose Top 150 terms for Subsumption
ParB	1 Term Per Segment, Choose Top 150 terms for Subsumption
ParC	1 Term Per Segment, Choose Top 300 terms for Subsumption

Table 2: Parameter Sets

4.2 Results

The main tests involved running our algorithm on the training sets with topic segmentation both enabled and disabled. In Table 3 we outline the percentage difference in the respective measure per test. From Table 3 we note that topic segmentation

Data Set	Max Dpt %	Max Docs %	Max Brd %	Avg Dpt %	Avg Doc %	Avg Nod %	Tree Msr %
ParA							
Reuters-A	3.33	31.41	-20.96	0.33	6.66	-24.69	6.29
Reuters-B	8.33	35.26	-19.49	1.09	1.24	-31.66	0.19
Reuters-C	0.00	-5.84	-13.54	0.69	-0.13	-37.71	89.09
ParB							
Reuters-A	1.72	22.64	-16.10	0.04	5.87	-27.06	5.88
Reuters-B	3.85	-7.73	-12.01	1.99	-6.25	-31.64	-8.00
Reuters-C	7.69	-8.50	-9.98	1.43	-7.20	-44.25	-8.22
ParC							
Reuters-A	0.00	3.62	-19.26	-5.34	18.65	-15.32	25.11
Reuters-B	0.00	-6.13	-12.57	-1.85	10.44	-11.71	12.56
Reuters-C	0.00	21.75	-12.15	1.00	-3.21	-20.56	-4.56

Table 3: Percentage Increase/Decrease in measures per test

allows a general mild increase in the average (and maximum) depth of the tree. This increase was not noted with the largest of the data set. This indicates that the parameters will require modification as the data set increases. We suspect, basing ourselves on other results (in [Mus05]) that to effect the depth we need to increase the top number of terms selected as input to subsumption.

The decrease in (node and document) breadth when topic segmentation is enabled indicates that our centroid selection is allowing clusters to be tighter and compact. This will result in clusters with documents which are much closer to the centroid and hence easier to browse for the user.

The general increase in *Tree.Msr* when topic segmentation is enabled indicates that the documents per level are improving. Although we do not want this value to be large, we do not want it to be very small either otherwise the tree would be large and sparse, and hence a burden on the browsing user.

5 Conclusion

Our research led us to the following contributions,

- An in-depth review of the various stages and approaches available implementing document clustering, mainly those applying statistical approaches.
- A review of evaluation methods applied to various clustering methods.
- The extraction of topics from documents by dividing text into segments and thereafter using them to build a hierarchy.
- A hierarchy evaluation technique which we were unable to apply due to computational issues related to the algorithm's time complexity.
- An evaluation technique extending the work in [SL00].

Our results are not as satisfying as expected. In general the generated tree depth is rather shallow for actual use. We believe the main reasons behind this are that our term selection policy together with the inherent characteristics of the subsumption algorithm which (by definition) takes in consideration only single terms. Results and tests are documented in [Mus05].

We have built an automatic document clustering framework which uses topic hierarchies as the main structure behind it, with no interaction by expert humans required for the process by harnessing the multi-topic properties of documents. Our approach though fails to produce a tree deep enough to make it useful in a real world scenario. Although we are sure this is possible, our term selection policies and intrinsic restrictions in the subsumption algorithm prevented us from producing a better quality tree (e.g. our tree is too shallow for practical use).

Further to that we supply an in-depth look at the applied evaluation methods and suggest one potential addition to the evaluation tools available and extend an existing [SL00] (and in our opinion, the most plausible of the lot) approach which does not depend on external test users and ready-made hierarchies to compare to.

This report is mainly an outline of our research related to automatic document clustering and hierarchy building. For an extensive, more in-depth, analysis and discussion please refer to [Mus05].

Acknowledgements. I would like to thank Dr. Chris Staff for his continuous and professional support, long discussions and constant feedback on my work. I would also like to thank the Department of Computer Science and Artificial Intelligence at the University of Malta for providing me with working space in order to keep my work focused and away from my living space (even though conceptually that also moved to the department for quite a while).

References

[Bil03] Philip Bille. Tree edit distance, alignment distance and inclusion. Technical report, IT University of Copenhagen, March 2003.

[BM98] L. Douglas Baker and Andrew K. McCallum. Distributional clustering of words for text classification. In W. Bruce Croft, Alistair Moffat, Cornelis J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 96–103, Melbourne, AU, 1998. ACM Press, New York, US.

[BYRN99] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press / Addison-Wesley, 1999.

[Cal94] J.P. Callan. Passage-Level Evidence in Document Retrieval. In W. Bruce. Croft and C.J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302 – 310, Dublin, Ireland, July 1994. Spring-Verlag.

[CKP93] Douglass R. Cutting, David Karger, and Jan Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 126–135, 1993.

[CKPT92] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM Press, 1992.

[DDL$^+$90] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[DFL$^+$88] Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Scott Deerwester, and Richard Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the Conference on Human Factors in Computing Systems CHI'88*, 1988.

[Hea] Marti A. Hearst. Texttiling: A quantitative approach to discourse segmentation. Technical Report S2K-93-24.

[Hea94] Marti A. Hearst. Multi-paragraph segmentation of expository text. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 9 – 16, New Mexico State University, Las Cruces, New Mexico, 1994.

[HP96] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 76–84, Zürich, CH, 1996.

[jKgL02] Han joon Kim and Sang goo Lee. An effective document clustering method using user-adaptable distance metrics. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 16–20. ACM Press, 2002.

[Jr.63] J.H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, (58):236–244, 1963.

[Kle98] Philip N. Klein. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms*, pages 91–102. Springer-Verlag, 1998.

[KTSK00] Philip Klein, Srikanta Tirthapura, Daniel Sharvit, and Ben Kimia. A tree-edit-distance algorithm for comparing simple, closed shapes. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 696–704. Society for Industrial and Applied Mathematics, 2000.

[Lan95] Ken Lang. News weeder: Learning to filter netnews. *ICML-95 (International Conference of Machine Learning)*, pages 331–339, 1995.

[Ler99] Kristina Lerman. Document clustering in reduced dimension vector space. http://www.isi.edu/∼lerman/papers/Lerman99.pdf (unpublished, last visited 09/02/2004), January 1999.

[LGXZ02] Xin Liu, Yihong Gong, Wei Xu, and Shenghuo Zhu. Document clustering with cluster refinement and model selection capabilities. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–198. ACM Press, 2002.

[MJK03] Bhushan Mandhani, Sachindra Joshi, and Krishna Kummamuru. A matrix density based algorithm to hierarchically co-cluster documents and words. In *Proceedings of the twelfth international conference on World Wide Web*, pages 511–518. ACM Press, 2003.

[MS00] Dharmendra S. Modha and W. Scott Spangler. Clustering hypertext with applications to web searching. In *Proceedings of the eleventh ACM on Hypertext and hypermedia*, pages 143–152. ACM Press, 2000.

[Mus05] Robert Muscat. Automatic document clustering using topic analysis. Master's thesis, University of Malta, Department of Computer Science and Artificial Intelligence, 2005.

[PL02] Patrick Pantel and Dekang Lin. Document clustering with committees. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 199–206. ACM Press, 2002.

[Rij99] C. J. Van Rijsbergen. *Information Retrieval.* 1999.

[SC99] Mark Sanderson and Bruce Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–213. ACM Press, 1999.

[Sel77] Stanley M. Selkow. The tree to tree editing problem. *Information Processing Letters*, 6(6):184–186, 1977.

[SKK00] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques, 2000.

[SL00] M. Sanderson and D. Lawrie. *In Croft, W. B Advances in Information Retrieval*, chapter Building, Testing, and Applying Concept Hierarchies. Kluwer Academic Publishers, 2000.

[SS97] H. Schutze and H. Silverstein. Projections for efficient document clustering, 1997.

[ST00] Noam Slonim and Naftali Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 208–215. ACM Press, 2000.

[Tai79] Kuo-Chung Tai. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, 1979.

[Ure00] Victoria Uren. An evaluation of text categorisation errors. In *Evaluation of Information Management Systems*, pages 78–87, 2000.

[Wal03] Matt Walker. Document cluster naming. Final Year Project at the The University of Texas at Austin, December 2003.

[WC00] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1103–1110, 2000.

[ZE98] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.

[ZE99] Oren Zamir and Oren Etzioni. Grouper: a dynamic clustering interface to Web search results. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1361–1374, 1999.

[ZK02] Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524. ACM Press, 2002.

[ZS89] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *Society for Industrial and Applied Mathematics, Journal on Computing*, 18(6):1245–1262, 1989.