Automatic Document Clustering Using Topic Analysis

Robert Muscat

Department of Computer Science and AI,
University of Malta

Abstract. Web users are demanding more out of current search engines. This can be noticed by the behaviour of users when interacting with search engines [12, 28]. Besides traditional query/results interactions, other tools are springing up on the web. An example of such tools includes web document clustering systems. The idea is for the user to interact with the system by navigating through an organised hierarchy of topics. Document clustering is ideal for unspecified search goals or for the exploration of a topic by the inexpert [21]. Document clustering is there to transform the current interactions of searching through a large amount of links into an efficient interaction where the interaction is navigation through hierarchies. This report will give an overview of the major work in this area, we will also propose our current work, progress and pitfalls which are being tackled.

1 Introduction

We propose document clustering as an alternative to traditional web searching where the user submits a query to a search engine and is provided a ranked list of documents from which he has to choose. The user traverses the list of links (usually aided by a description which is usually a document snippet for each link).

With document clustering a query is optional, and initially the user is presented with an overall structure of the topics related to the domain. Selecting a particular topic from those provided will reveal further topical subdivisions which intuitively reside beneath the selected topic e.g. selecting *science* will reveal *mathematics, physics, computing, chemistry* together with other documents which fit under *science* but which do not fit under any other subcategory in *science*.

A number of terms are used to describe document clustering including text categorisation or document classification. Even though the results of each process are in the same spirit described above, the approach to solving the problem is different. Categorisation/classification methodologies use a machine learning (statistical approach) where a learning algorithm is applied on a manually built data set to allow the algorithm to find patterns within the data which allow classification. We consider manually built data sets an issue due to the fact that typically indexers are not able to agree on the majority of their indexing decisions [28, 30]. Typical text/document clustering on the other hand uses text analysis and models to decide how documents are grouped. Both these methods have their own pros and cons which we will illustrate.

CATEGORISATION

Pros
⇒ The structure of the topics within documents is known before hand.
⇒ Straight forward to implement after the documents have been appropriately vectorised. A number of accepted vectorising methods is already available.
⇒ Typically, machine learning systems make common assumptions across all data set, which is not always correct.
Cons
⇒ Requires a training period which may be lengthy at times.
⇒ Selection of training data is crucial.
⇒ Adding new topics to the collection requires reconfiguration of the training data and retraining.

CLUSTERING

Pros
⇒ The structure is generally detected automatically, without (or with minimal) human input.
⇒ Does not require any previous knowledge, the approach extracts all required information from the domain.
Cons
⇒ The structure is usually less intuitive and semantically nodes are less related than when manually built.
⇒ Requires a lot of processing.
⇒ Assumption are not made, but everything is seen from the model's perspective.

Throughout this report we will assume some knowledge of basic models used in information retrieval. We follow with a small introduction to clustering methodologies (domain independent discussion). The rest of the report describes work directly related to document clustering/classification, followed by evaluation methodologies and proposed techniques.

1.1 Data Clustering

The main types of clustering algorithms are *agglomerative* and *divisive*. As the name suggests, the former assumes that a domain of elements are initially separate clusters, during the clustering process elements are combined into clusters until finally all elements are in one all inclusive cluster. Divisive on the other hand is a top down approach, starting with one large cluster and ending up with each element in a separate clusters.

The principle agglomerative algorithms are *single link* [27], *complete link*[6] and *average link*[31]. On the assumption that initially all elements are clusters, each algorithm compares all clusters together to decide which will be joined (to form a single cluster) by comparing each element of the cluster. The two clusters with the minimum distance are then joined. The distance varies depending on the algorithm used:

Single-Link The distance between two clusters is the *minimum* distance between any two elements from the two clusters.
Complete-Link The distance between two clusters is the *maximum* distance between any two elements from the two clusters.

Average-Link The distance between two clusters is the *average* distance between all elements from the two clusters.

The above algorithms, by nature, are a traversal across dichotomies. To reach a particular number of clusters it is required to traverse all stages before reaching a target number of clusters. In most applications this might be an overhead and alternatives to this approach have been suggested.

One such alternative is *k-means* [22]. The advantages are that the algorithm allows k clusters to be generated immediately (k is the function parameter). Initially k random elements are selected and assumed to be centroids. Centroids are points in the cluster which are the closest to all other elements in the cluster. All others elements are assigned to the nearest centroids and a new centroid is recalculated. The process is reiterated until no further elements move from one cluster to another. Other variations on this algorithm exist [14, 17, 19].

2 Related Work

In this section we review some of the major approaches document clustering relates to our approach. Some common evaluation methodologies will also be discussed.

2.1 Document Clustering

Scatter/Gather [10] was one of the first attempts to group document 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 joined and reclustered using another algorithm *Buckshot* (which is faster but less accurate). By iterating this process and selecting different clusters each time the user is shown different views of the domain [4, 10, 3]. 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 higher up in the hierarchy).

Another important approach was that taken Zamir and Etzioni [32, 33]. 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 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 spare the user from seeing duplicate data. Finally, the set of available graphs are scored and the top ten are returned to the user. Grouper [33] was an online application of the *suffix tree algorithm* and used results from *HuskySearch* [33] a meta-search engine which was under development by the same team.

Other work by Lerman [19] and Schütze and Silverstein [25], takes a more standard approach to clustering but concentrate, on the preprocessing of the document space. They reduce the dimensions

of documents 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 (Latent Semantic Indexing [7, 5], discussed later) has generally improved performance of information retrieval system. Lerman [19] shows that LSI does in fact improve cluster quality. Her work shows that for this to happen, the dimension size selected must be optimal (or near optimal). [19] 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.

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 beforehand, 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 must be labelled a priori.

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. Refer to [20] for a review. 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 promoted to replace the `Reuters 21578` data set.

The closest method which is relevant to our requirements is that used by Mandhani et al. in [23]. 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 approach which can be applied to automatic document clustering. However, this approach generates a large number of results (separate values per level). An obvious enhancement would integrate all these separate results in fewer (ideally one).

3 Proposed Approach

In this section we propose our approach to document clustering. A number of techniques borrowed from other research areas in information retrieval to propose a method with which we can automatically generate clusters and topic hierarchies.

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[2] http://about.reuters.com/researchandstandards/corpus/

3.1 Clustering Methodology

Our document clustering solution is based on the assumption that a document can refer to more than one topic/concept. We will use these concepts to build the hierarchy and assign each document to the part of the tree which relates mostly to it.

The first challenge involves subdividing each document into topics. A number of topic extraction algorithms exist which help detect shifts in the topic of discussion in the text stream. A number of techniques are already available with the two main ones being *TextTiling* [8, 9, 11] and *C99* [2].

TextTiling determines topic shifts by first assuming that the document is divided up into blocks. Usually these blocks are taken to be paragraphs or blocks of n terms. `TFxIDF` (Term Frequency and Inverse Document Frequency) is calculated across the blocks (assumed to be separate documents) and the terms in each block. After each term is assigned a weight, pairs of sequential blocks of k terms are taken and compared using a distance algorithm. This will generate a sequence of distances for all pairs. This sequence is then smoothed using two techniques, one based on an average similarity value and another using simple median algorithm.

These sections will be analysed and keywords will be extracted from each section. The terms to select will require the analysis of the term weightings which have been assigned globally. We need to investigate other term weightings schemes to see which weightings best help determine the topic terms. Currently the suggested weightings are `TFxIDF`, document frequency and term frequency. We will also suggest local term and block frequency but these latter have not been tested yet. This phase is important and from preliminary tests we noticed that it can determine the quality of the overall system, since the quality of the tree is determined by the terms selected.

When a number of terms are selected they are used to build a concept hierarchy using specialised algorithms. These algorithms analyse the relations between terms and determine which term, semantically, lies beneath another. The idea is that of detecting relations of sub-topics, sub-categories and co-similarity e.g. "physics" and "nuclear physics", "science" and "physics", and "chemistry" and "physics". Given a set of such relations, we then build a full fledged topic hierarchy which is what the user will be navigating through at the very end. Documents are assigned to the topics which are deemed most similar to the topic in the hierarchy.

A number of algorithms exist which detect topic relationship. Besides Lawrie *et al.* which bases his approach on the *Dominating Set Problem* [18] we will describe the work by Sanderson and Croft in [24] where they describe the subsumption algorithm. The algorithm analyses the co-occurrence of a particular term A with term B. To find a relation we assume that A subsumes B if $P(A|B) = 1$ and $P(B|A) < 1$.

By applying these ideas we claim that we will be able to automatically build a document hierarchy, with clusters containing documents which discuss similar topics. With this approach, we will not be able to compare with statistical/machine learning techniques but at least we generate an acceptable tree which can be traversed by a user.

All our comparisons will be performed using Latent Semantic Indexing (LSI) which is a technique used to map a term-document index into a different (usually smaller to make comparisons faster) space. LSI maps the index into a smaller dimension whilst retaining the information in the original index [7, 5].

The final step would be to assign a naming to each cluster by looking at its contents. We intend to extract phrases which are common inside each document in the cluster and use that as a cluster identifier.

3.2 Evaluation

As we have discussed in Section 2.2 evaluation is an issue which has been tackled a number of times but not yet settled, since the standards available are only useful for statistical/machine learning approaches. We suggest a method for evaluating automatic document clustering where the hierarchical structure is not known a priori.

We propose three test bases to be able to evaluate the quality of document hierarchies,

Tree Distance we will be using a tree distance algorithm in order to calculate the edit distance between the tree from the data set and the generated hierarchy. The edit distance is a measure which is based on a weight scheme assigned to each operation (insertions, deletions and edits) which transforms tree A into tree B. A number of algorithms exist which carry out this task. Zhang and Shasha's algorithm returns the minimum edit distance between two trees [34].
Two tests will use the tree edit distance.
- An overall tree edit distance which will measure the two trees as a whole.
- An iterative test where starting from the leaves cluster nodes are replaced iteratively with the cluster content moving up in the hierarchy. For each iteration the tree distance is calculated.

More work related to tree edit distances is found in [1, 15, 16, 29, 26, 13].

Cluster Quality Measures This stage will measure cluster cohesiveness and quality. These measures are internal measures and they only base their calculations on the contents of the cluster without comparing it to any external source of information (like the original data set).

Recall and Precision are the standard measure used in information retrieval. They will be used to test each cluster. An overall average can be used to measure the overall system *recall* and *precision*. Refer to Section 2.2 for details.

Note that since our clusters will not be labelled, we will be matching clusters between source and generated clusters by taking the two with the shortest distance between them.

4 Results

At the moment no results are available. Zhang and Shasha's algorithm resulted in extreme computation times especially since the algorithm requires a regular expression to find patterns in strings representing the trees. We are at the moment trying to find a way to speed up this process. We are also considering a tree edit distance algorithm which does not guarantee a *minimum* edit distance but does allow us to compare tree structures in tractable times.

This hurdle stopped the development process of our ideas until we have fixed our evaluation algorithms. At the moment we are concentrating our work on the evaluation process to get it up and running.

5 Conclusion

We do not expect our results to compare with statistical/machine learning approaches since the hierarchical structure is manually constructed beforehand. Our work should, however be able to automatically build structures which are navigable by the average user. We intend to show this by doing a user study to determine navigability.

The most critical stage in the process is the term selection to automatically identify topic terms. Part of the research will be dedicated to the fine-tuning of the current models to enable better term selection. This model is also critical in identifying cluster titles.

At this stage work is in progress to make the evaluation process as fast, accurate and easy as possible to allow future reference to our work and allow use of the same evaluation models we suggest in our system.

References

1. Philip Bille. Tree edit distance, alignment distance and inclusion. Technical report, IT University of Copenhagen, March 2003.
2. Freddy Y. Y Choi. Advances in domain independent linear text segmentation. In *Proceedings of NAACL-00*, 2000.
3. 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.
4. 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.
5. 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.
6. D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, (20):346–366, 1977.
7. 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.
8. Marti A. Hearst. Texttiling: A quantitative approach to discourse segmentation. Technical Report S2K-93-24.
9. 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.
10. 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.
11. Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 59–68. ACM Press, 1993.
12. Bernard J. Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998.
13. Taeho Jo. Evaluation function of document clustering based on term entropy. In *The Proceedings of 2nd International Symposium on Advanced Intelligent System*, pages 302–306, 2001.
14. J.H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, (58):236–244, 1963.
15. 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.
16. 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.
17. Jouko Lampinen. On clustering properties of hierarchical self-organizing maps. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks, 2*, volume II, pages 1219–1222, Amsterdam, Netherlands, 1992. North-Holland.

18. Dawn Lawrie, W. Bruce Croft, and Arnold Rosenberg. Finding topic words for hierarchical summarization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 349–357. ACM Press, 2001.
19. Kristina Lerman. Document clustering in reduced dimension vector space. http://www.isi.edu/ lerman/papers/Lerman99.pdf (last visited 09/02/2004), January 1999.
20. David D. Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 246–254. ACM Press, 1995.
21. 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.
22. J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Symp. Math. Statist. and Probability, 5th, Berkeley*, volume 1, pages 281–297, Berkeley, CA, 1967. AD 66981. 157., 1967. Univ. of California Press.
23. 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.
24. 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.
25. H. Schutze and H. Silverstein. Projections for efficient document clustering, 1997.
26. Stanley M. Selkow. The tree to tree editing problem. *Information Processing Letters*, 6(6):184–186, 1977.
27. R. Sibson. Slink: an optimally efficient algorithm for a complete link method. *The Computer Journal*, (16):30–34, 1973.
28. Associate Professor Amanda H. Spink, S. Ozmutlu, and H. C. Ozmutlu. A day in the life of web searching: An exploratory study. *Information Processing and Management 40.2*, pages 319–45, 2004.
29. Kuo-Chung Tai. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, 1979.
30. D. Tarr and Harold Borko. Factors influencing inter-indexing consistency. In *Proceedings of the American Society for Information Science (ASIS) 37th Annual Meeting*, volume 11, pages 50–55, 1974.
31. E. M. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing & Management*, 22(6):465–476, 1986.
32. Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.
33. 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.
34. K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.