Journey to the Centre of the Star: Various Ways of Finding Star Centers in Star Clustering

Derry Tanti Wijaya and Stéphane Bressan*

School of Computing, National University of Singapore, 3 Science Drive 2, Singapore
derrytan@comp.nus.edu.sg, steph@nus.edu.sg

Abstract. The Star algorithm is an effective and efficient algorithm for graph clustering. We propose a series of novel, yet simple, metrics for the selection of Star centers in the Star algorithm and its variants. We empirically study the performance of off-line, standard and extended, and on-line versions of the Star algorithm adapted to the various metrics and show that one of the proposed metrics outperforms all others in both effectiveness and efficiency of clustering. We empirically study the sensitivity of the metrics to the threshold value of the algorithm and show improvement with respect to this aspect too.

1 Introduction

Clustering is the task of grouping similar objects. In a graph, when vertices represent objects and (possibly weighted) edges represent similarity among objects, clustering is the separation of dense from sparse regions where clusters are the dense regions. A vector space clustering naturally translates into a graph clustering problem for a dense graph in which vertices correspond to vectors and pairs of vertices are connected with an edge whose weight is the cosine of the corresponding vectors. The graph is a clique [1]. In 1998, Aslam et al. [2] proposed the Star algorithm for graph clustering and gave a complete presentation in 2004 in [3], in which they proposed both an off-line and an on-line version and studied analytically and empirically the properties and performance of their algorithms. Star algorithm replaces the computation of vertex covering of a graph by cliques by a very simple computation of dense sub-graphs: lower weight edges of the graph are first pruned; then vertices with higher degree are chosen in turn as Star centers while vertices connected to a center become satellites. The algorithm terminates when every vertex is either a center or a satellite; each center and its satellites forming one cluster. Two critical elements in this algorithm are the threshold value for the pruning of edges and the metrics for selecting Star centers. In their paper [2], Aslam et al. derived the expected similarity between two satellite vertices in a cluster as a function of similarities between satellites and their star center and the threshold value. They show empirically that this theoretical lower bound on the expected similarity is a good estimate of the actual similarity. Yet the metrics they

* This work was funded by the National University of Singapore ARG project R-252-000-285-112.

R. Wagner, N. Revell, and G. Pernul (Eds.): DEXA 2007, LNCS 4653, pp. 660–670, 2007.

propose for selecting star centers: the degree of a vertex; does not leverage this finding. Our work is motivated by the suspicion that degree is not the best possible metrics for selecting Star centers. A metric that maximizes intra cluster similarity, i.e. average similarity among all pairs of vertices in the cluster, should improve performance. We propose a series of novel, yet simple, metrics for selecting Star centers that take into account the similarity between vertices (i.e. weight of edges). In particular we propose average metrics which, as we argue analytically, maximizes intra-cluster similarity. We empirically study the performance of off-line and on-line versions of Star algorithm adapted to the various proposed metrics. We show that average metrics outperforms all others in both effectiveness and efficiency. Our contribution is the presentation of metrics and their comprehensive comparative performance analysis with real world and standard corpora for the task of clustering documents.

2 Background and Related Works

2.1 Vector and Graph Clustering

Algorithms for vector and graph clustering can be grouped into: partitioning, hierarchical, and graph algorithms. Partitioning algorithms, such as K-means [4] divide vertices into K clusters by defining K centroids and associate each vertex to nearest centroid. The algorithm iteratively recalculates the position of K centroids until the centroids no longer move. Hierarchical algorithms [5] can be categorized into agglomerative and divisive. The agglomerative method treats each vertex as a separate cluster, iteratively merges clusters that have smallest distance from one another until all clusters are grouped into one, yielding a hierarchical tree of clusters. Divisive method is the reverse of agglomerative that starts with all objects in one cluster and divides them into smaller clusters. Markov Clustering (MCL) is a graph clustering algorithm based on simulation of (stochastic) flow/random walks in graphs [6]. The Star clustering algorithm replaces the NP-complete computation of a vertex-cover by cliques by the greedy, simple and inexpensive computation of star shaped dense sub graphs. Star clustering does not require the indication of an a priori number of clusters. It also allows the clusters to overlap. Star clustering analytically guarantees a lower bound on the similarity between objects in each cluster and computes more accurate clusters than either the single [7] or average [8] link hierarchical clustering.

2.2 Star Clustering and Extended Star Clustering

To produce reliable document clusters of similarity σ (i.e. clusters where documents have pair-wise similarities of at least σ, where σ is a user-defined threshold), Star algorithm starts by representing the document collection by its σ-similarity graph where vertices correspond to documents and there is an undirected edge from vertex v_i to vertex v_j if their cosine similarity in a vector space [1, 9] is greater than or equal to σ. Star clustering formalizes clustering by performing a minimum clique vertex cover with maximal cliques on this σ-similarity graph [10]. Since covering by cliques is an NP-complete problem [11, 12], Star clustering approximates a clique cover greedily

by dense sub-graphs that are star shaped, consisting of a single Star center and its satellite vertices, where there exist edges between the Star center and each satellite vertex. Star clustering guarantees pair-wise similarity of at least σ between the Star and each of its satellites. However, it does not guarantee such similarity between satellite vertices. By investigating the geometry of the vector space model, Aslam et al. derive a lower bound on the similarity between satellite vertices and predict that the pair-wise similarity between satellite vertices in a Star-shaped sub-graph is high. Together with empirical evidence [2], Aslam et al. conclude that covering σ-similarity graph with Star-shaped sub-graphs is an accurate method for clustering a set of documents.

Each vertex v in Star clustering has a data structure containing v.degree: the degree of v, v.adj: the list of v's adjacent vertices, v.marked: a bit denoting if v is already in a cluster, and v.center: a bit denoting if v is a Star center. The off-line Star algorithm (for static data) sorts vertices by degree. Then, it scans the sorted vertices from highest to lowest degree as a greedy search for Star centers. Only vertices that are not yet in a cluster can be Star centers. Once a new Star center v is selected, v.center and v.marked bits are set, and for all vertices w adjacent to v (i.e. $w \in v.adj$), w.marked is set. Only one scan of the sorted vertices is needed to determine all Star centers. Upon termination when all vertices have their marked bits set, these conditions must be met: (1) the set of Star centers are the Star cover of the graph, (2) a Star center is not adjacent to any other Star center, and (3) every satellite vertex is adjacent to at least one center vertex of equal or higher degree. The algorithm has a run time of $\Theta (V + E\sigma)$ where V is the set of vertices and Eσ edges in the σ-similarity graph Gσ [2].

Star algorithm has some drawbacks that the Extended Star algorithm [13] by Gil et al. proposes to solve. The first drawback [13] is the Star covers (hence the clusters produced) are not unique. When there are several vertices of the same highest degree, the algorithm arbitrarily chooses one as Star. The second drawback is because no two Star centers can be adjacent to one another; the algorithm can produce illogical clusters [13]. The extended Star algorithm addresses these issues by choosing Star centers independently of document order. It uses complement degree of a vertex v, CD(v), which is the degree of v only taking into account its adjacent vertices not yet included in any cluster: $CD (v) = | v.adj \setminus Clu |$; where Clu is the set of vertices already clustered. Extended Star algorithm also considers a new notion of Star center. A vertex v is considered a Star if it has at least an adjacent vertex x with less or equal degree than v, and x satisfies these conditions: (1) x has no adjacent Star, or (2) the highest degree of Stars adjacent to x is not greater than v's degree. Extended Star algorithm has two versions: unrestricted and restricted. In the restricted version, another condition is imposed: only unmarked vertices: vertices not yet included in any cluster can be Star centers. Extended Star processes vertices independently of document order. If more than one vertex with the same highest degree exists, the extended Star selects all of them as Star centers. Extended Star algorithm allows two Stars to be adjacent to one another as long as they satisfy the required conditions to be Star centers.

On-line Star algorithm [2] supports insertion and deletion of vertices from the graph. When a new vertex is inserted into (or deleted from) the graph, new Stars may need to be created and existing Stars may need to be destroyed. On-line Star algorithm maintains a queue containing all satellite vertices that have the possibility of being 'promoted' into Star centers. As long as these vertices are indeed satellites, the existing Star cover is correct. The vertices in the queue are processed in order of their

degree (from highest to lowest). When an enqueued satellite is promoted to Star center, one or more existing Stars may be destroyed; creating new satellites that have the possibility of being promoted. These satellites are put into the queue and the process repeats. Our discussion in the next section is, to some extent, orthogonal to the improvement of the extended Star algorithm or to the extensions of the on-line version. Our algorithms retain the logic of the off-line and on-line extended and original Star algorithms but aim within reasonable complexity at improving the performance of off-line, on-line, extended and original Star algorithms by maximizing the 'goodness' of the greedy vertex cover through novel yet simple metrics for selecting Star centers.

3 Finding Star Centers

Our work is motivated by the suspicion that degree is not the best metrics for selecting Star centers. We believe Star centers should be selected by metrics that take into consideration the weight of the edges in order to maximize intra-cluster similarity.

3.1 Markov Stationary Distributions

Star centers are 'important' or 'representative' vertices. Intuitively, it seems that maximum flow vertices as used in Markov clustering should be good candidate Star centers. This idea is similar to the one used in Google's page Rank algorithm [14] for quantifying the importance of a Web page. The first metrics that we propose is therefore the probability of reaching a vertex after a random walk, i.e. in the stationary distribution of the Markov matrix. A Markov matrix $A=[a_{ij}]$ is a square matrix such that: $\forall i, \forall j\ 0 \le a_{ij} \le 1$, and $\forall i\ \Sigma_{\forall j}\ a_{ij} \le 1$ (or $\forall j\ \Sigma_{\forall i}\ a_{ij} \le 1$). The adjacency matrix of a weighted graph is or can be normalized into a symmetric Markov matrix. We represent the similarity graph (without threshold) G = (V, E) as an adjacency matrix A where each entry aij is the normalized similarity between vertex i and j in G: aij = sim (i, j) / $\Sigma_{\forall x \in V}$ sim (i, x). A is therefore a sub-stochastic matrix. Each entry a_{ij} represents the probability of passing from i to j traversing one edge during a random walk in G. The matrix A×A or A^2, represents probabilities of two edge transversals. At infinity, the Markov chain A^k converges. We call A* the fix point of the product of A with itself. The sum of the stationary values is the probability to end up on a given vertex after a random walk. For a given vertex represented by row and column i in the adjacency matrix A of the graph, it is the value $\Sigma_{\forall j}\ b_{ij}$ in the matrix $A^*= [b_{ij}]$. Because A is a Markov matrix A* can be computed directly as follows: $A^* = (I-A)^{-1}$.

Incorporating this metrics in Star algorithm, we sort vertices by the sum of their stationary values and pick unmarked vertex with highest value to be the new Star center. In the remainder we refer to Star algorithm in which Star centers are determined using sum of stationary values as Star-markov: the off-line star algorithm with Markov stationary distribution metrics. The computation of A* is relatively expensive (it is accounted for in experimental results presented in section 4). For this reason, we do not consider this metrics for extended and on-line versions of the algorithm; neither do we investigate in this paper technique for on-line approximation of the value.

3.2 Lower Bound, Average and Sum

In their derivation of expected similarity between satellite vertices in a Star cluster [2], Aslam et al. show that the similarity $\cos(\gamma_{i,j})$ between two satellite vertices v_i and v_j in a Star cluster is such that:

$$\cos(\gamma_{i,j}) \geq \cos(\alpha_i)\cos(\alpha_j) + (\sigma / \sigma + 1)\sin(\alpha_i)\sin(\alpha_j) . \quad (1)$$

Where $\cos(\alpha_i)$ is the similarity between the Star center v and satellite v_i and $\cos(\alpha_j)$ is the similarity between the Star center v and satellite v_j. They show that the right hand side of inequality (1) above is a good estimate of its left hand side. Hence it can be used to estimate the average intra-cluster similarity. For a cluster of n vertices and center v, the average intra-cluster similarity is therefore: $(\Sigma_{(vi, vj) \in v.adj \times v.adj} (\cos(\gamma_{i,j}))) / n^2 \geq ((\Sigma_{vi \in v.adj} \cos(\alpha_i))^2 + (\sigma / \sigma + 1)(\Sigma_{vi \in v.adj} \sin(\alpha_i))^2)/n^2$; where $\gamma_{i,j}$ is the angle between vertices v_i and v_j and α_i is the angle between v and vertex v_i and where v.adj is the set of vertices adjacent to v in Gσ (i.e. vertices in the cluster). This is computed on the pruned graph (i.e. the σ-similarity graph). Based on this average intra-cluster similarity, we derive our first metrics: for each vertex v in Gσ, we let: $lb(v) = ((\Sigma_{vi \in v.adj} \cos(\alpha_i))^2 + (\sigma / \sigma + 1)(\Sigma_{vi \in v.adj} \sin(\alpha_i))^2)/n^2$. We call this metrics **lb(v)**, the lower bound. This is the theoretical lower bound on the actual average intra-cluster similarity when v is Star center and v.adj are its satellites. The lower bound metrics is slightly expensive to compute because it uses both cosine and sine. We consider two additional simpler metrics and empirically evaluate whether they constitute good approximations of lb(v). The metrics are computed on the pruned graph. Namely, for each vertex v in Gσ, we let: $ave(v) = \Sigma_{\forall vi \in v.adj} \cos(\alpha_i)/$ degree(v) and $sum(v) = \Sigma_{\forall vi \in v.adj} \cos(\alpha_i)$; where α_i is the angle between v and vertex v_i. We call the metrics **ave(v)** and **sum(v)** the average and sum metrics, respectively. Notice that the ave(v) metrics is the square root of the first term of the lb(v) metrics. lb(v) grows together with ave(v). Therefore ave(v) should be a criteria equivalent to lb(v) for the selection of Star centers. Incorporating these metrics in off-line and on-line Star algorithms, we sort vertices by sum and average and pick unmarked vertex with highest sum and average to be the new Star center, respectively. In the remainder we refer to Star algorithm in which Star centers are determined using the sum and average as the Star-sum and Star-ave, respectively. We incorporate the lower bound, average and sum metrics in the original and extended Star algorithms by using lb(v), sum(v) and ave(v) in the place of degree(v) and by defining and using complement lower bound **CL(v)**, complement sum **CS(v)**, and complement average **CA(v)** in the place of complement degree, CD(v), respectively. We define $CL(v) = ((\Sigma_{vi \in v.adj \setminus Clu} \cos(\alpha_i))^2 + (\sigma / \sigma + 1)(\Sigma_{vi \in v.adj \setminus Clu} \sin(\alpha_i))^2)/n^2$, $CS(v) = \Sigma_{\forall vi \in v.adj \setminus Clu} \cos(\alpha_i)$, $CA(v) = \Sigma_{\forall vi \in v.adj \setminus Clu} \cos(\alpha_i)/$ CD(v), where Clu is the set of vertices already clustered.

We integrate the above metrics in the Star algorithm and its variants to produce the following extensions: (1) Star-lb: the off-line star algorithm with lb(v) metrics; (2) Star-sum: the off-line star algorithm with sum(v) metrics; (3) Star-ave: the off-line star algorithm with ave(v) metrics; (4) Star-markov: the off-line star algorithm with Markov stationary distribution metrics; (5) Star-extended-sum-(r): the off-line restricted extended star algorithm with sum(v) metrics; (6) Star-extended-ave-(r): the

off-line restricted extended star algorithm with ave(v) metrics; (7) Star-extended-sum-(u): the off-line unrestricted extended star algorithm with sum(v) metrics; (8) Star-extended-ave-(u): the off-line unrestricted extended star algorithm with ave(v) metrics; (9) Star-online-sum: the on-line star algorithm with sum(v) metrics; (10) Star-online-ave: the on-line star algorithm with ave(v) metrics. For the sake of simplicity and concision, we only show results for lower bound in the original Star algorithm in the performance analysis section (we do not evaluate Star-extended-lb-(u) and Star-extended-lb-(r), the off-line unrestricted and restricted extended star with lower bound metrics, nor Star-online-lb: the on-line star algorithm with lower bound metrics).

4 Experiments

In order to evaluate the proposed metrics, we compare the performance of our extensions with the original off-line and on-line Star clustering algorithms and with the restricted and unrestricted extended Star clustering algorithms. We use data from Reuters-21578 [15], TIPSTER–AP [16] and our original collection: Google. The Reuters-21578 collection contains 21,578 documents that appeared in Reuter's newswire in 1987. The TIPSTER–AP collection contains AP newswire from the TIPSTER collection. Our original collection: Google contains news documents obtained from Google News website [17] in December 2006. Each collection is divided into several sub-collections. By default and unless otherwise specified, we set the value of threshold σ to be the average similarity of documents in the given sub-collection. We measure effectiveness (recall, r, precision, p, and F1 measure, $F1 = (2 * p * r) / (p + r)$), efficiency (running time) and sensitivity to σ. In each experiment, for each topic, we return the cluster which "best" approximates the topic, i.e. cluster that produces maximum F1with respect to the topic: $topic(i) = \max_j \{F1(i, j)\}$; where F1(i, j) is the F1 measure of the cluster number j with respect to topic i. The weighted average of F1 measure for a sub-collection is calculated as: $F1 = \Sigma (n_i/S) * F1(i, topic(i))$; for $0 \leq i \leq N$; where N is the number of topics in the sub-collection, n_i is the number of documents belonging to topic i in the sub-collection, and $S = \Sigma n_i$; for $0 \leq i \leq N$. For each sub-collection, we calculate the weighted-average of precision, recall and F1-measure produced by each algorithm. We then present the average results over each collection.

4.1 Performance of Off-Line Algorithms

We empirically evaluate the effectiveness (precision, recall, F1 measure) and efficiency (time) of our proposed off-line algorithms: Star-markov, Star-lb, Star-sum, Star-ave and compare their performance with the original Star algorithm: Star and its variant: Star-random that picks star centers randomly. The results reported for Star-random are average results for various seeds that gives us a base line for comparison.

In Fig. 1 we see that Star-lb and Star-ave achieve the best F1 values on all collections. This is not surprising because as we argued, lb(v) metrics maximizes intra-cluster similarity. The fact that our Star-ave achieves comparable F1 to Star-lb is evidence that average is a sufficient metric for selecting star centers. In Fig. 1, based on

F1: when compared to original Star, our proposed algorithms: Star-ave and Star-markov by far outperform the original Star on all collections. An interesting note is that Star-random performs comparably to original Star when threshold σ is the average similarity of documents in the collection. This further proves our suspicion that degree may not be the best metric. In Fig. 2 we see that our proposed algorithms with the exception of Star-markov perform as efficiently as original Star. Star-markov takes longer time as it uses matrix calculation to compute random walk. As expected, Star-lb takes the longest time due to its expensive computation.

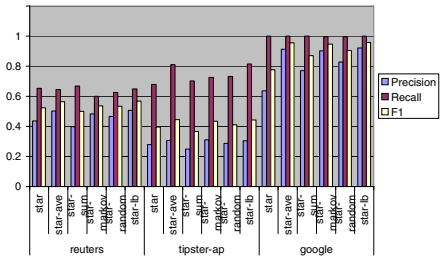

Fig. 1. Effectiveness of off-line Star

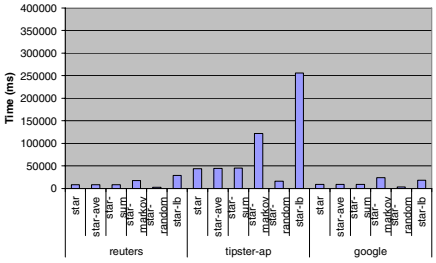

Fig. 2. Efficiency of off-line Star

From Fig. 1 and Fig. 2 we can conclude that (1) since Star-random is more efficient and achieves comparable F1 to original Star, using degree to pick stars may not be the best metric, (2) since Star-ave is more efficient and achieves comparable F1 to Star-lb; Star-ave can be used as a good approximation to Star-lb to maximize the resulting intra-cluster similarity.

4.2 Order of Stars

We empirically demonstrate that Star-ave indeed approximates Star-lb better than other algorithms by a similar choice of star centers.

In Fig. 3 and 4 we present the order in which the algorithms choose their star centers on TIPSTER-AP and Reuters collections (similar trend is observed on Google

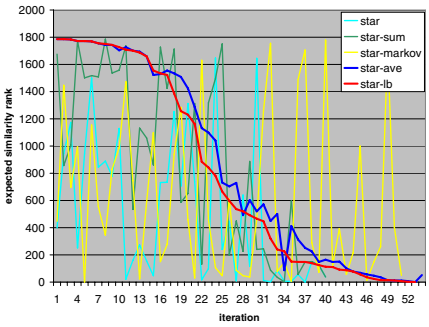

Fig. 3. Order of Stars for TIPSTER-AP

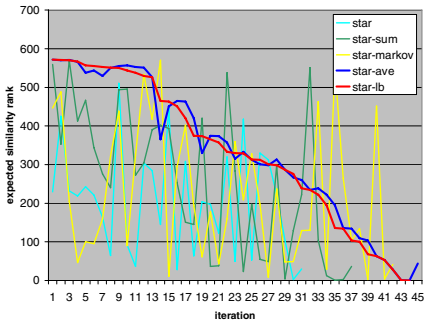

Fig. 4. Order of Stars for Reuters

collection): from the first star center to the n^{th} star center and where star centers are ranked by their expected intra-cluster similarity (computed from equation 1) from highest to lowest. Star-lb chooses star centers in the order of their expected similarity rank, from highest to lowest. In Fig. 3 and 4, we see that Star-ave chooses star centers in an order similar to Star-lb. This is not the case of the other algorithms. Therefore picking star centers in descending order of expected intra-cluster similarity can be approximated simply by picking star center in descending order of average similarity with its adjacent vertices.

4.3 Performance of Off-Line Algorithms at Different Threshold (σ)

Star clustering has one parameter σ. With a good choice of σ, just enough edges are removed from the graph to disconnect sparsely connected dense sub-graphs. Removing too few edges will group these sparsely connected sub-graphs together; producing high recall but low precision clusters. Removing too many edges will break these dense sub-graphs into smaller, perhaps not-so-meaningful components; producing low recall but high precision clusters. Fig. 5 illustrates the empirical performance evaluation of our proposed off-line algorithms at different threshold values σ on Reuters collection (similar observation is found on TIPSTER-AP and Google collections). For a similarity graph G (V, E), we represent σ as a fraction (s) of the average edge weight (E_{mean}) in G, i.e. $\sigma = E_{min} + (s * E_{mean})$; where E_{min} is the minimum edge weight in G.

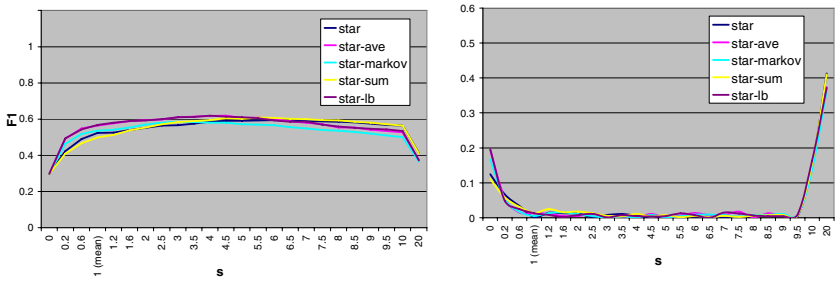

Fig. 5. Performance of off-line algorithms with varying σ on Reuters data

In Fig. 5, we see that Star-ave and Star-markov converge to a maximum F1 at a lower threshold than the original Star. Star-ave and Star-markov are able to 'spot' sparsely connected dense sub-graphs and produce reliable vertex cover even when there are few edges removed from the graph. On the contrary, the original Star algorithm needs to remove more edges from the graph before it is able to produce reliable clusters. The maximum F1 value of Star-ave is also higher than the maximum of the original Star. In Fig. 5 we see that the F1 values of Star-ave coincide closely with the F1 values of Star-lb at all thresholds. This is further evidence that Star-ave can be used to approximate Star-lb at varying thresholds. In Fig. 5, we see that the F1 gradient (the absolute change in F1 value with each change in threshold) is smaller (some approaching zero) for Star-ave and Star-markov as compared to the original Star. This

small gradient means the value of F1 does not change/fluctuate much with each change in threshold. The smaller F1 gradient means that Star-ave and Star-markov are less sensitive to the change in threshold as compared to the original Star.

From Fig. 5 we can conclude that: (1) Based on maximum F1 value: our proposed algorithm Star-ave outperforms the original Star, (2) our proposed algorithms: Star-ave and Star-markov are able to produce reliable clusters even at a lower threshold where there are only fewer edges removed from the graph; (3) F1 values of Star-ave coincide closely with F1 values of Star-lb at all thresholds; this is further evidence that Star-ave can approximate Star-lb at any given threshold value.

4.4 Performance of Off-Line Extended Star Algorithms

We present the results of the experiment that incorporates our idea into the extended Star. In Fig. 6 and 7, we present effectiveness and efficiency comparison between: our algorithm: Star-ave that has performed the best so far, the original restricted extended star: Star-extended-(r), our algorithms: Star-extended-ave-(r), and Star-extended-sum-(r). Due to space constraints we do not present the results of comparison with the unrestricted version of extended star in which we observe similar findings.

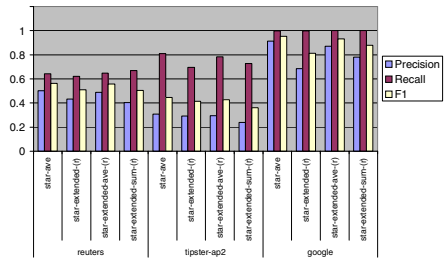

Fig. 6. Effectiveness of restricted extended Star

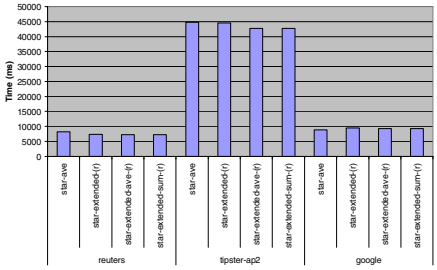

Fig. 7. Efficiency of restricted extended Star

In Fig. 6, based on F1 values: we see that our proposed algorithm: Star-ave; outperforms Star-extended-(r) on all collections. This is despite the fact that Star-ave uses only very simple idea to incorporate into the original Star. Our proposed algorithm: Star-extended-ave-(r) that incorporates the idea of complement-ave metric to the extended Star algorithms improves the performance of Star-extended-(r). In Fig. 7, in terms of efficiency: our proposed algorithms perform comparably or faster than the extended Star; with the exception on TIPSTER-AP collection (in which Star-ave takes longer time). We believe this difference in time could be because Star-ave picks different Star centers from the extended Star. We also see that incorporating the idea of complement-ave and complement-sum to extended Star does not reduce its original efficiency on all collections. From Fig. 6 and 7, we can conclude that: (1) our proposed algorithm: Star-ave obtain higher F1 values than the extended Star on all collections, (2) incorporating the idea of complement-ave metric to the extended Star improves its F1 without affecting its efficiency on all collections.

4.5 Performance of On-Line Algorithms

Fig. 8 and 9 illustrate the effectiveness and efficiency of the on-line algorithms: the original on-line Star: Star-online, and our on-line algorithms, Star-online-ave and Star-online-sum. We compare the performance of these algorithms with a randomized version of the algorithm: Star-online-random that picks star centers randomly.

Star-online-ave outperforms Star-online on all collections in terms of F1 (cf. Fig. 8). Both Star-online-ave and Star-online-sum perform better on all collections than Star-Random. They are more efficient (cf. Fig. 9) than Star-online on TIPSTER-AP data and comparable on other collections. From Fig. 8 and 9, we can conclude that Star-online-ave achieves higher F1 than Star-online. However, due to the fact that Star-online-ave may pick different star centers than Star-online; its efficiency may be affected.

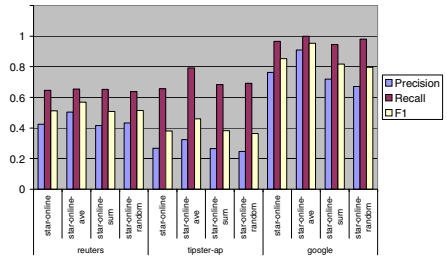

Fig. 8. Effectiveness of on-line algorithms

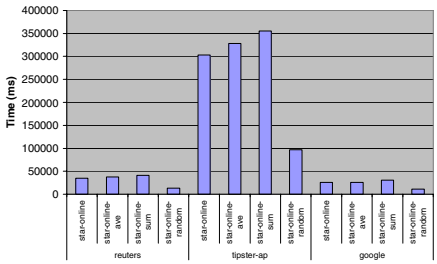

Fig. 9. Efficiency of on-line algorithms

5 Conclusion

We suspected that the metrics used for selecting star centers in Star clustering is not optimal. The theoretical argument was presented in the original papers of Star clustering but was not exploited. We therefore proposed various new metrics for selecting star centers: Markov metrics that find vertices of maximum flow; and metrics estimating and commensurating to the maximum intra-cluster similarity (lower bound, average and sum). We empirically studied the performance of off-line star, extended restricted and unrestricted Star and on-line Star with these different metrics. Our results confirm our conjecture: selecting star centers based on degree (as proposed by the original algorithm inventors) performs almost as poorly as a random selection. One needs to use a metrics that maximizes intra-cluster similarity such as the lower bound metrics. While it indeed yields the best results, it is expensive to compute. The average metrics is a fast and good approximation of the lower bound metrics in all variants of Star algorithm: (1) Star-ave yields up to 4.53% improvement of F1 with a 19.1% improvement on precision and 1.77% on recall; (2) Star-extended-ave-(r) yields up to 14.65% improvement of F1 with a 27.2% improvement on precision and 0.25% on recall; (3) Star-extended-ave-(u) yields up to 138% improvement of F1 with a 102% improvement on precision and 3.4% on recall; (4) Star-online-ave yields up to an outstanding 20.81% improvement of F1 with a 20.87% improvement on precision and 20.67% on recall. We notice that, since intra-cluster similarity is maximized, it is

precision that is mostly improved. We can therefore propose Star-online-ave as a very efficient and very effective graph clustering algorithm.

References

1. Salton, G.: Automatic Text Processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley (1989)
2. Aslam, J., Pelekhov, K., Rus, D.: Static and Dynamic Information Organization with Star Clusters. In Proceedings of the 1998 Conference on Information Knowledge Management, Baltimore, MD (1998)
3. 3. Aslam, J., Pelekhov, K., Rus, D.: The Star Clustering Algorithm. In Journal of Graph Algorithms and Applications, 8(1) 95–129 (2004)
4. MacQueen, J. B.: Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. Berkeley, University of California Press, 1:281-297 (1967)
5. Johnson, S. C.: Hierarchical Clustering Schemes. Psychometrika, 2:241-254 (1967)
6. van Dongen, Stijn Marinus: Graph clustering by flow simulation - Tekst. - Proefschrift Universiteit Utrecht (2000)
7. Croft, W. B.: Clustering large files of documents using the single-link method. Journal of the American Society for Information Science, 189-195, November 1977
8. Voorhees, E.: The cluster hypothesis revisited. In Proceedings of the 8th SIGIR, 95-104
9. Salton, G.: The Smart document retrieval project. In Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, 356-358
10. Karp, R.: Reducibility among combinatorial problems. Computer Computations, 85–104, Plenum Press, NY (1972)
11. Lund, C., Yannakakis, M.: On the hardness of approximating minimization problems. Journal of the ACM 41, 1960-981 (1994)
12. Press W., Flannery B., Teukolsky S., Vetterling W.: Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press (1988)
13. García, R.J. Gil, Contelles, J.M. Badía, Porrata, A. Pons: Extended Star Clustering Algorithm. Proc. Of CIARP'03, LNCS 2905, 480-487 (2003)
14. Brin Sergey, Page Lawrence: The anatomy of a large-scale hypertextual Web search engine. Proceedings of the seventh international conference on World Wide Web 7, 107-117 (1998)
15. http://www.daviddlewis.com/resources/testcollections/reuters21578/ (visited on December 2006)
16. http://trec.nist.gov/data.html (visited on December 2006)
17. Google News (http://news.google.com.sg)