Distributed Pivot Clustering with Arbitrary Distance Functions

L. Karl Branting
7525 Colshire Drive
McLean, Virginia
USA
lbranting@mitre.org

Abstract—

This paper describes an algorithm, Distributed Pivot Clustering (DPC), that differs from prior distributed clustering algorithms in that it requires neither an inexpensive approximation of the actual distance function nor that pairs of elements in the same cluster share at least one exact feature value. Instead, DPC requires only that the distance function satisfy the triangle inequality and be of sufficiently high-granularity to permit the data to be partitioned into canopies of optimal size based on distance to reference elements, or *pivots*. An empirical evaluation demonstrated that DPC can lead to accurate distributed hierarchical agglomerative clustering provided that the triangle inequality and granularity requirements are met.

INTRODUCTION

Clustering very large data sets is essential for data-mining tasks in many diverse domains, including many forms of textual, visual, sequential, spatial, and temporal analysis. A key challenge of hierarchical clustering of very large datasets is avoiding calculation of a global distance table, an $\Omega(n^2)$ operation that is intractable for large data collections.

In domains in which members of the same cluster can be guaranteed to share a common feature value, distance calculations can be restricted to pairs sharing at least one such feature. In the general case, however, there may be no single criterion for distinguishing pairs belonging to the same cluster from those that do not, other than the distance function itself. For example, cross-document named-entity resolution typically entails clustering within-document coreference chains having complex contextual information. No single coreference-chain feature is either necessary or sufficient for such pairs of coreference chains to denote the same entity.

An intractable global distance table calculation can be obviated if comparisons involving each element can be restricted to a subset of instances that are plausible candidates for being in the same cluster. This can be achieved if the original dataset can be inexpensively partitioned into subsets, possibly overlapping, of elements that are not too dissimilar from one another under the distance function.

Techniques for partitioning instances given only a distance function that satisfies the triangle inequality have been known for decades. For example, Burkhart-Keller Trees [1], [2] facilitate indexing by partitioning instances by distance from *pivots*, *i.e.,* randomly selected reference instances. If the distance function satisfies the triangle inequality, pairs of instances that differ by more than k in distance to a pivot cannot differ from each other by less than k. The distance between such pairs need not be calculated if k is an upper limit on the distance that can separate members of the same cluster. Similarly, FastMap [3] projects a metric space onto a Euclidean space based on relative distance to reference elements. A variety of other pivot-based indexing techniques exploiting this insight are discussed in [4].

The insight that distances to reference points can be used to filter pairs that must be very dissimilar and therefore need not be compared was applied to clustering in *canopy clustering* [5]. As originally described, canopy clustering uses two distance functions: a "cheap and approximate" function based on an inverted index; and a second more expensive and accurate function. Each canopy consists of instances within some threshold T_1 of a random point (*i.e.*, a pivot) under the inexpensive function, and those within $T_2 < T_1$ are restricted from being in any other canopy. Clustering using the expensive, precise function is limited to the contents of each canopy. For agglomerative clustering, this procedure reduces the cost of a global distance table to the sum of the costs of building distance tables for the contents of each canopy.

The procedure described in the original canopy clustering paper demonstrated the utility of partitioning instances into overlapping subsets based on distance to pivots, but this approach is not applicable in the general case of an arbitrary distance function for which no approximation, such as an inverted index, is available. This dependence on an approximation to the actual distance function is also found in more recent approaches to large-scale clustering, which typically depends on the assumption that all pairs of entities in the same cluster must share an exact match on at least one feature, such as a name [6].

In the absence of an inexpensive approximate distance function based, for example, on exact feature matches, canopies must be constructed using the precise distance function itself. In place of an approximate distance *function*, an approximate *threshold*, T', must be identified that will guarantee that every cluster will be contained within a canopy of radius T' around some instance (if this condition cannot be met, no set of pivots can lead to the correct clusters).

Under these circumstances, that is, when which there is no inexpensive approximation to the distance function, any practical solution to the general clustering problem must solve the following three problems:

1) Finding an appropriate approximate threshold T' for the given distance function. If T' is too large, canopies will be too large to cluster tractably, *e.g.*, Hadoop processing nodes will time-out for lack of progress. T' values that are too small give rise to boundary effects, described in more detail below, which can cause suboptimal clusters.
2) Finding an appropriate number of pivots or, equivalently, canopies. Too many canopies degrade performance by causing redundant clustering. Too few canopies, like too small T's, result in boundary effects.
3) When overlapping canopies cause a set of instances to be clustered multiple times in separate clusters, assigning a single unique cluster assignment for all the instances. If elements of multiple identical canopies are not given a single unique cluster assignment, they will appear to be in separate clusters, degrading clustering accuracy.

Choosing an appropriate value for T' is by far the most complex of these three tasks. This paper sets forth an approach to clustering in Hadoop that solves these three problems in a manner that is agnostic as to both the particular clustering algorithm applied at each node and the distance function, requiring only that the distance function satisfy the triangle inequality.

Motivation and Task Description

Large-scale clustering problems with complex distance functions are typified by cross-document entity resolution (CDER) [7], [8], [9], [10]. *Entity resolution* comprises the subtasks of *entity matching* (*i.e.*, recognizing that two distinct name strings may denote the same person, and *entity disambiguation* (*i.e.*, recognizing that two identical name string instances may denote different people) [11]. In a typical approach to CDER, entity resolution is first performed on individual documents, producing *entity records* each of which contains a coreference chain consisting of "mentions" of a single individual in a single document. Entity records may also contain additional information associated with entity mentions, such as relationships to co-occurring entities and concepts expressed in the text surrounding the entity. Entity records are then clustered based on a distance function defined over coreference chains and other single-document record information.

Determining the similarity between pairs of entity records can require complex reasoning. For example, if "John Doe" and "Jonathan A. Doe" are authors of the same book, this is strong evidence that they are the same person, whereas having the same nationality is weak evidence. Sharing an uncommon name is much stronger evidence of denoting the same person than is sharing a common name. Because of the complexity of evaluating the similarity of pairs of entity records, entity records often contain information that is not amenable to representation as feature vectors. As a result, in some cases it is not feasible to model the distance function as Euclidean distance in feature space. Cross-document entity resolution is particularly important and challenging in large-scale document collections, such as the English Wikipedia or the entire World Wide Web, which can only be stored and processed in a distributed fashion.

If a threshold T can be identified such that any two records, R_i and R_j that differ by no more than T (*i.e.*, $d(R_i, R_j \leq T$), probably denote the same individual, then cross-document entity resolution can be performed by a distributed partitional clustering procedure that does the following:

- Given:
  - A set of records R in a distributed file system
  - A distance function d over pairs $(R_i, R_j), R_i, R_j \in R$ such that
    - d satisfies the triangle inequality
    - $\exists$ some threshold T such that if $d(R_i, R_j) \leq T$, then R_i and R_j should belong to the same cluster.
- Do:
  - Find a smallest partition P of R such that $\forall p \in P(R_i \in p \wedge R_j \in p \rightarrow d(R_i, R_j) \leq T)$

The contribution of this paper is to demonstrate how pivot-based techniques can be used to implement this distributed partitional clustering procedure in the MapReduce framework to accurately cluster distributed datasets even in the absence of an approximate distance function. Specifically, this paper shows how to find the optimal approximate threshold T', determine an appropriate number of pivots, and assign members of overlapping canopies to a single unique cluster. An empirical evaluation demonstrates that the accuracy of this approach depends critically on aspects of the distance function, including (1) satisfaction of the triangle inequality, (2) granularity, and (3) the dimensionality of the data to which the distance function applies.

Distributed Pivot Clustering (DPC)

Distributed pivot Clustering (DPC) is an approach to clustering very large data collections in a MapReduce environment. DPC takes as input a set of records, R, and a distance function, d, together with three additional parameters (1) an upper limit S on the number of instances that can be tractably clustered on a single node, (2) a threshold T representing the upper limit on distance between two equivalent records, and (3) the target degree of overlap between canopies, O. S, T, and O can be determined empirically by clustering a subset of data on a single node: S through timing tests, and T and O through cross validation, if ground-truth data is available.

As with other pivot-based algorithms, DPC exploits the fact that, for functions satisfying the triangle equality, instances that are close to the same pivot must also be close to each other. At a high level, the DPC approach is as follows:

- Calculate a value for T' such that the number of elements within T' of each pivot *i.e.,* the mean canopy size C, will be as close as possible to S, that is, each canopy will consist of the maximum number of elements that can be tractably clustered at a single node.

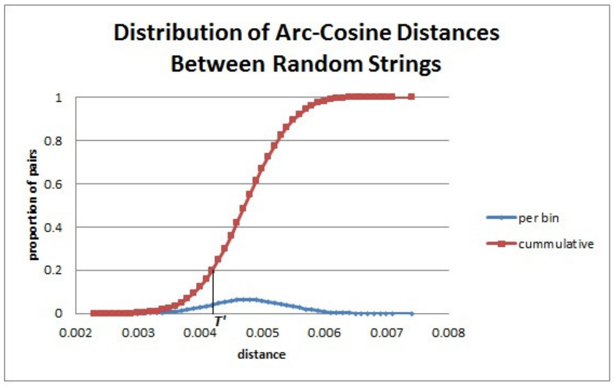


Fig. 1. A histogram of per-bin and cumulative pair-wise distance counts for 100,000 sample pairs, 100,000 random 80-character strings, character-vector arc-cosine distance, and 1,000 bins.

- Select a set of P random pivots from the unclustered records such that $P * C = |R| * O$, that is, so that the number of pivots times the canopy size, $P*C$, is equal to total number of elements times the target overlap, $R*O$.
- Construct a canopy for each pivot p consisting of all unclustered records r such that $d(p,r) \leq T'$. The triangle inequality guarantees that the distance between each pair of records in a given canopy will be no more than $2 * T'$.
- Cluster the contents of each canopy using threshold $T \leq T'$. Select a unique cluster assignment for instances belonging to multiple clusters.
- Repeat this procedure while any unclustered records remain.

Calculating T'

The purpose of the first two MapReduce jobs is to calculate a value for T' that is large enough to minimize boundary effects but small enough to be tractable. The complex nature of many distance functions means that pairwise distances are, in general, not normally distributed. Instead, the distribution is best modeled as a histogram of counts.

The first MapReduce step samples random pairs of instances and writes out, for each node, the frequency counts of each distinct distance at that node ([key: binNumber, value: count]), where the default number of bins is 1,000,000. The second MapReduce job sums these values across all nodes, producing a histogram for the sampled instances of the entire set.

For example, Figure 1 is a histogram of per-bin and cumulative pair-wise distance counts for 100,000 sample pairs taken from 100,000 random 80-character strings, where distance is measured as the arc-cosine between pairs of character-vectors. The vertical axis is the proportion of pairs falling within each bin shown in the horizontal axis. Specifically, the value of bin k is the number of pairs R_i, R_j for which $(k-1)/b \leq d(R_i, R_j) < k/b$.[1]

[1]No bin is needed for the distance 1.0 because pairs with maximum dissimilarity should never be in the same canopy.

Based on the histogram, T' is calculated as

$$T' = \arg\max_{T'} (\sum_{k=1}^{\lfloor b*T' \rfloor} bin_k / |sample| \leq S/|R|)$$

That is, T' is the inverse cumulative sum of the proportion of the entire sample in each bin of the histogram. The calculation is performed by simply iterating through the bins of the histogram starting with the bin corresponding to the lowest distance and stopping when the sum of histogram counts reaches the largest sum less than or equal to $|sample| * S/|R|$ (since a larger value might lead to canopies too large to tractably cluster at each node). In Figure 1, T' has been selected such that the cumulative distribution is 0.2. Each canopy that includes all records within this T' of a pivot can be expect to contain about 20% of the entire data set. To select canopies with a smaller proportion of the entire data set, which is typical necessary for large data sets, a much smaller value for T' would be selected.

Pivot Selection

The third MapReduce job selects P random pivots from the unclustered records, where $P = |R| * O/S$, that is, so that the number of pivots is equal to total number of elements times the target overlap over the target canopy size. To insure that each node has exactly the same pivots, the pivots are sent to the distributed cache at the conclusion of this job.

Canopy Construction and Clustering

The fourth mapper compares each unclustered record to each pivot. Those that match a pivot are written to the canopy for that pivot ([key: pivot, value: record]). The fourth reducer then clusters the elements of each canopy, *i.e.,* clusters all records that match a common pivot, writing the results as [key: record, value: clusterSummary]. Each clusterSummary records the number of elements of the cluster and the mean similarity of the key to the other elements of the cluster, information that is used in the unique cluster assignment step. Pivots that match no other records are treated as singleton clusters.

Unique Cluster Assignment

The Fifth MapReduce job starts with an empty Mapper. The Reducer iterates over pairs of the form [key: record, value: clusterSummary]. Each such pair represents an alternative assignment of the record to a different cluster, each corresponding to a separate canopy in which the key occurred (there will be a separate canopy for each pivot p for which $d(p,s) \leq T'$). Each record is assigned to single cluster according to the following procedure:

- Select the largest cluster containing the record.
- If there are multiple clusters of the same size, select the cluster for which the mean distance between the record and the elements of the cluster is least.
- If there are still multiple clusters, impose an arbitrary ordering of clusters (*e.g.*, select the cluster with minimum hash code) to break the symmetry in a consistent way.

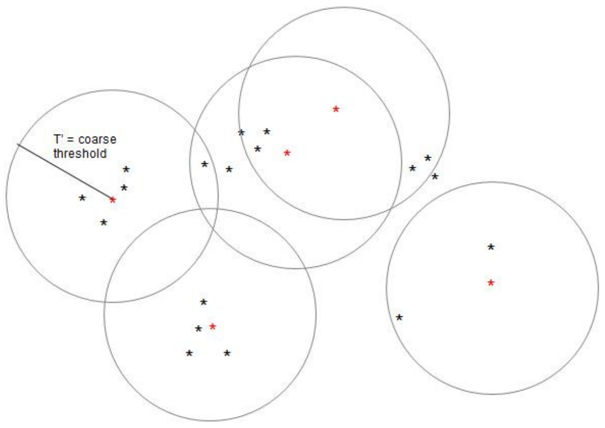


Fig. 2. Canopies, indicated by large circles, consist of all instances within T' of a pivot, shown in red.

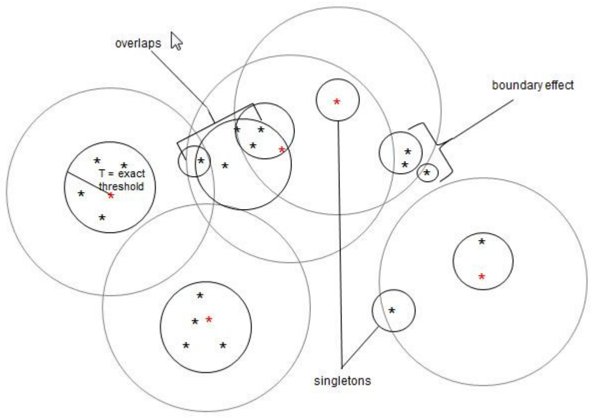


Fig. 3. Canopy overlaps and singletons do not affect accuracy, but boundary effects degrade accuracy.

Segregation

Two additional MapReduce jobs are needed to identify the records from the original set, R, that are not yet clustered (because they did not match any pivot), and to move any such records to a separate directory for the next iteration of the algorithm.

Recalculation of Statistics

If instances remain to be clustered, which can happen if the original data set does not equal the union of the canopies, the sequence of MapReduce steps is repeated, including calculation of the distribution of pair-wise distances. Recalculation of distribution statistics is necessary because at each iteration unclustered instances are more likely to be in sparsely populated regions than those clustered. Thus, the distribution typically changes each distribution.

Overlaps, Singletons, and Boundary Effects

Figure 2 illustrates notionally how canopies (depicted as large circles) consist of all instances within T' of a pivot (shown in red). Since pivots are chosen at random, the canopies can overlap in unpredictable ways. Not shown in this figure is the fact that instances are initialed distributed unpredictably among the nodes in the Hadoop cluster, so that instances that are close together under the distance function are not, in general, initially on the same node.

Intuitively, the purpose of the pivots is identify instances that are close enough together under the distance function that they can be clustered on a single node. The fourth mapper, described above, indexes all such instances with a common key so that they are all handled by the same reducer, and the fourth reducer clusters all instances in each canopy.

Figure 3 illustrates that when canopies are clustered, the result may include singleton or overlapping clusters, and that there may be boundary effects in which a ground-truth cluster only partially overlaps any canopy. Overlapping clusters don't reduce accuracy as long as instances are consistently assigned to a unique largest and best matching cluster, as per the procedure above.

Boundary effects, however, degrade clustering accuracy. A ground-truth cluster that is not completely contained within some canopy can not be accurately detected. This is the motivation for requiring the degree of canopy overlap to be as large as tractable to minimize the likelihood of boundary effects.

Two Ways a Distance Function Can Affect Clustering Accuracy

There are many ways to measure the distance between pairs of entities, such as co-reference chains, for the purpose of clustering. However, the choice of distance function can have a significant effect on the effectiveness and accuracy of distributed pivot clustering. As shown above, the triangle inequality guarantees that each pair of records in a canopy differ by no more than $2 * T'$. If the triangle inequality is not satisfied, this guarantee may fail, meaning that canopies are no longer guaranteed to consist of similar records. For example, suppose that $2 * T < T'$ and there is some record r_1 that is less than T from both pivot V and from a second record, r_2. If the triangle inequality is satisfied, record r_2 can differ from a pivot V by no more than $2 * T < T'$ and will therefore be in the same canopy as r_1. If the triangle inequality is not guaranteed, r_2 could be arbitrarily far from V and therefore fail to be in V's canopy. Under these circumstances, r_1 and r_2 could fail to be in the same cluster.

There is a second, less obvious, scenario in which a distance function can preclude accurate clustering even if the distance function satisfies the triangle inequality. DPC calculates T' based on the histogram of pairwise distances in the corpus. If the distance function d has too few distinct values, then there may be no possible values for T' such that (1) $T' \geq T$ (necessary to avoid boundary effects) and (2) the cumulative sum of T' is no larger than $|sample| * S/|R|$, that is, the T' neighborhoods of pivots contain no more than the upper limit, S, on tractable canopy size. For example, if d is normalized

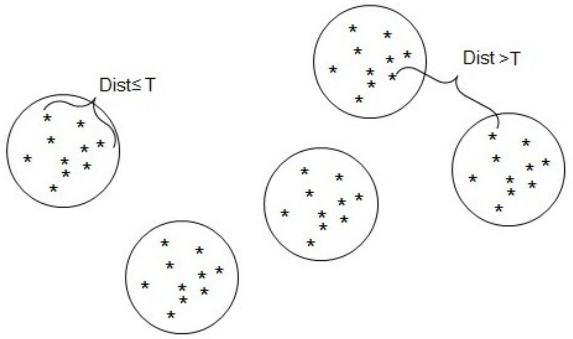
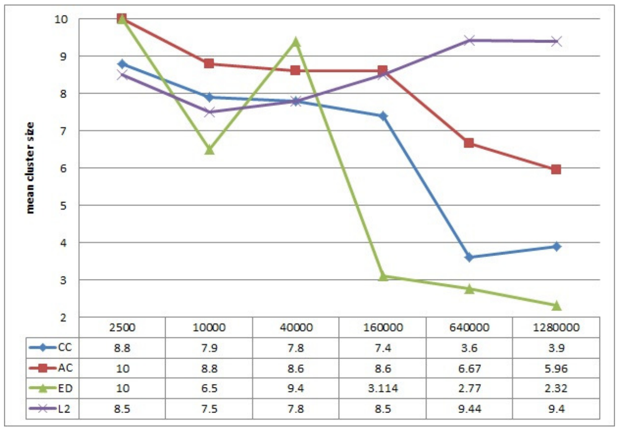


Fig. 4. Artificial data sets in which members of the same cluster differ by at most T, while elements of different cluster differ by strictly greater than T.



	2500	10000	40000	160000	640000	1280000
CC	8.8	7.9	7.8	7.4	3.6	3.9
AC	10	8.8	8.6	8.6	6.67	5.96
ED	10	6.5	9.4	3.114	2.77	2.32
L2	8.5	7.5	7.8	8.5	9.44	9.4

Fig. 5. Mean recovered cluster size for clusters with actual size 10 and overlap of 4.

edit distance,[2] the number of distinct values of d can be no greater than the length of the longest string being compared. If the number of records being clustered is large in proportion this number of distinct values, it may be impossible to satisfy both conditions (1) and (2) above.

In sum, pairs of instances may fail to be clustered if either (1) the distance function does not guarantee the triangle inequality, in which case instances close to each other may not be close to the same pivot, or (2) the distance function has very few possible values relative to the number of instances being clustered, in which case there may be no possible value of the distance function that can produce canopies small enough to be tractable but big enough to avoid boundary effects.

Experimental Evaluation

To demonstrate empirically the effect of distance function characteristics on clustering accuracy, an evaluation was performed using artificial data on which four distinct distance functions were defined:

1) Character cosine (CC). Instances are strings of characters (default length 80), and distance is the cosine of the frequency vectors of the character strings. CC does not guarantee the triangle inequality.
2) Arc-cosine of character cosine (AC). Identical to CC except that the cosine is converted to an angle by arc-cosine function. AC guarantees the triangle inequality.
3) Edit distance (ED). Normalized edit distance between two character strings (default length 80) consisting of edit distance divided by the length of the longest of the two strings being compared (by default, both strings are the same length). ED satisfies the triangle inequality but has a limited number of possible values.
4) Euclidean distance (L2). Instances are vectors of real number, of length 4 by default.

[2]Normalized edit distance is edit distance divided by the length of the longest of the two strings being compared. Hereafter in this paper, the term "edit distance" is used as shorthand for "normalized edit distance".

To simplify evaluation of clustering accuracy, artificial data sets were constructed such that every cluster had the same number n of elements (defaulting to 10), the maximum distance between any pair of elements in each cluster was T, and the minimum distance between any pair of elements not in the same cluster was strictly greater than T, as depicted in Figure 4. The clustering algorithm applied to each canopy was complete (maximum) linkage agglomerative hierarchical clustering, with a maximum pairwise distance within each cluster of T. Under this approach, if every cluster c discovered by the algorithm is of size $|c| = n$, then the correct structure has been discovered. This is because the clustering procedure guarantees that no cluster can contain any pair whose dissimilarity is greater than T. This precludes clustering errors in which pairs that should be in separate clusters are incorrectly placed in the same cluster. The only possible clustering error is separation of the instances that should be in one cluster into two or more clusters, *i.e.,* the only possible error is false negatives. Thus, a mean cluster size of less than n indicates a loss of accuracy from failure to cluster entities that differ by less than T.

In the first experiment, data sets were created of sizes 2,500, 10,000, 40,000, 160,000, 640,000, 1,280,000, the Euclidean data was 4-dimensional, the CC, AC, and ED data consisted of length-80 character sequences, the target canopy size was 2,000, and target overlap, O, was 4.

The results, shown in Figure 5, illustrate the effects of errors of both types described above. The vertical axis of Figure 5 represents the mean size of the clusters returned by DPC on each data set. Mean size of 10 means that the exact cluster size has been recovered, while lower values mean that instances that differ by less that T were not clustered together.

For data sets of all sizes, AC had a higher mean cluster size, and therefore greater accuracy, than CC even though both the underlying character strings themselves and their representation as character frequency vectors was identical. Replacing a function that violated the triangle inequality (character cosine)with one that preserved it (arc cosine) consistently improved accuracy.

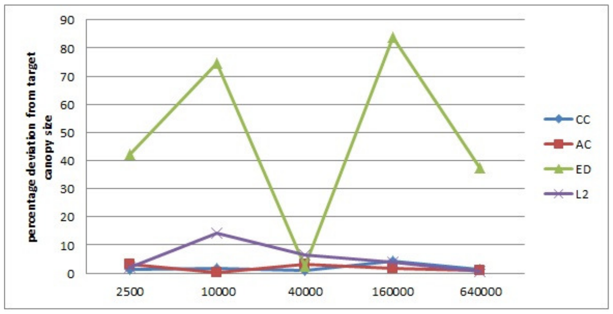


Fig. 6. Mean error rate in canopy size.

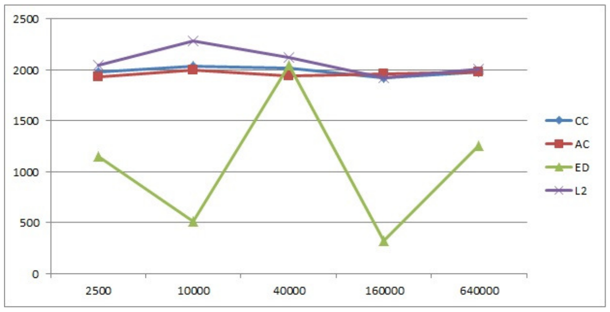


Fig. 7. Mean canopy size.

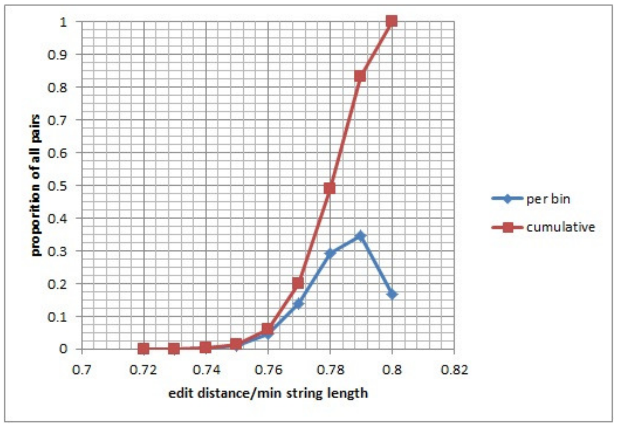


Fig. 8. Per bin and cumulative proportion of all pairs for edit distance.

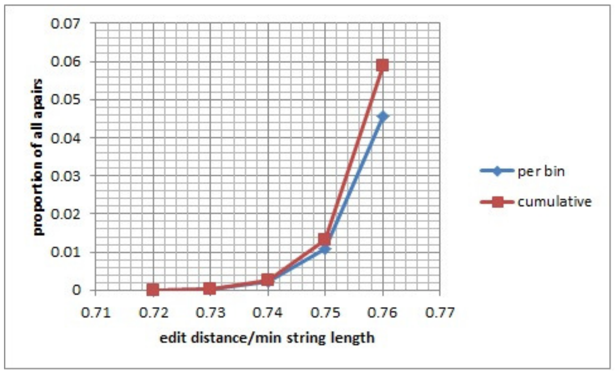


Fig. 9. Per bin and cumulative proportion of all pairs for edit distance.

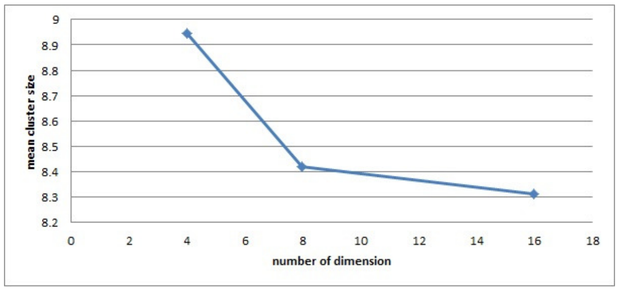


Fig. 10. Accuracy of L2 as a function of the number of dimensions (40,000 instances).

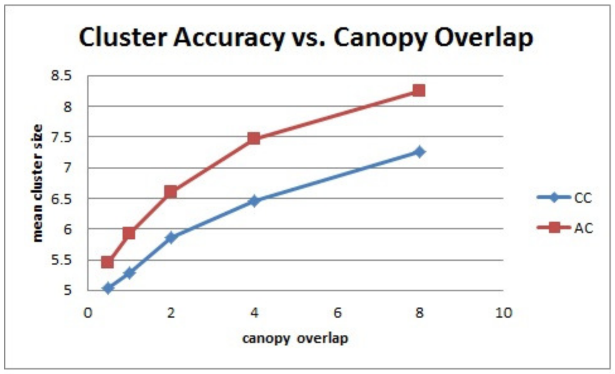


Fig. 11. Accuracy of AC and CC as a function of target overlap, O, for 160,000 instances.

ED satisfies the triangle inequality,[3] but Figure 5 shows that DPC accuracy plummeted for ED as the data set size grew.

Figure 6 shows that ED is the only distance function for which there is a high error rate in canopy size, and Figure 7 shows that the error rate is the result of the canopy size being much lower than the target of 2,000 for every size data set except 40000.

To understand why the limited number of possible values degrades the accuracy of canopy-sizes in large data sets, consider that for the largest set in the evaluation above, 1,280,000, a single canopy of size 2,000 is 2,000/1,280,000 = 0.0015625 of the total number of instances being clustered. Accordingly, T' should be selected so that the cumulative sum of histogram bins is 0.0015625. Figure 8 shows the very small number of distinct bins in the histogram for 100,000 sample pairs drawn from 100,000 random 80 character strings. Figure 9 shows that the cumulative proportion increases from 0.00036 to 0.00252 as edit distance increases from 0.73 to 0.74 (*i.e.*, the number of character differences increases by 1). Choosing T' to be 0.73 would lead to canopies containing on average 0.00252 of the total, or roughly 3226 per canopy, whereas $T' = 0.74$ leads to canopies on average of 0.00036, or roughly 461 per canopy, small enough to lead to boundary errors. Since DPC selects T'

[3]Edit distance satisfies the triangle inequality because edit-distance(A,C) can never exceed edit-distance(A,B) + edit-distance(B,C), that is, the length of the shortest sequence of edits from A to B to C is an upper bound on the length of the shortest sequence of edits from A to C.

such that canopies will be no greater than the target size, S, T' is selected to be 0.74, leading to the under-sized canopies and resultant severe boundary effects shown above.

Occasional accurate performance under edit distance, such as occurred with the 40,000 instances data set in the example above, results when one of the small number of possible distance values happens to coincide with the proportion of instances required for the target canopy size. The insufficient granularity of edit distance stands in contrast to the situation shown in Figure 1, in which the granularity of character-vector arc-cosine is effectively limited only by the number of bins.

For larger data sets, clustering was considerably more accurate under Euclidean distance than under arc-cosine distance. One possible explanation for this phenomenon is that the higher dimensionality of character vectors degrades performance, since higher dimensions lead, all else being equal, to less distance variance and therefore diminished discrimination among instances based on distance [12]. Figure 10 provides confirmation for this hypothesis because it shows that the accuracy under Euclidean distance diminishes with increasing dimensions.

The effect of varying the target canopy overlap, O, is shown in Figure 11. Increasing target overlap increased cluster accuracy over the entire range tested. This suggests that a high level of canopy overlap is needed to minimize boundary effects.

In summary, the results shown in Figure 5 illustrate two ways in which an inappropriate distance function can lead to degraded clustering accuracy: the function may violate the triangle inequality, like character cosine, or the function may have too few distinct values to permit the set of instances to be partitioned into canopies of the right size for distributed clustering, like edit distance. Euclidean distance suffers from neither of these defects, and Figure 5 shows that it supports accurate clustering even at large scales for four-dimensional vectors, although accuracy decreases for higher dimensions. Character vector arc cosine also satisfies both criteria, but its high dimensionality appears to cause some loss of discrimination based on distance to pivots and therefore some loss of clustering accuracy.

Conclusion

This paper has described a procedure, DPC, for tractable clustering of large data collections under the MapReduce framework. In contrast to previous large-scale clustering algorithms, DPC does not depend on the existence of an inexpensive approximation of the actual distance function nor on the requirement that pairs of elements in the same cluster share at least one exact feature value. Instead, DPC depends only on a distance function that satisfies the triangle inequality and that is sufficiently high-granularity to partition the data set into partitions small enough to cluster on a single processor. This paper has shown that key technical challenge of DPC, determining a value of T' that produces canopies of appropriate size, can be solved by sampling the pairwise distances distribution in the data set. The experimental evaluation suggests that performance is better with lower-dimensional than higher-dimensional data and is improved by significant canopy overlap.

DPC demonstrates that very large data sets data can be tractably and accurately clustered in the MapReduce framework, for any distance function with sufficient granularity that satisfies the triangle inequality. This result should extend the range of domains amenable to large-scale hierarchical clustering beyond those for which simplifying assumptions, such as that all pairs of elements in the same cluster necessarily share at least one common feature value.

Acknowledgment

This work was funded under contract number CECOM W15P7T-09-C-F600. The MITRE Corporation is a not-for-profit Federally Funded Research and Development Center chartered in the public interest.

References

[1] W. A. Burkhard and R. M. Keller, "Some approaches to best-match file searching," *Commun. ACM*, vol. 16, no. 4, pp. 230–236, 1973.

[2] R. A. Baeza-Yates and G. Navarro, "Fast approximate string matching in a dictionary," in *String Processing and Information Retrieval*, 1998, pp. 14–22.

[3] C. Faloutsos and K.-I. Lin, "Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '95. New York, NY, USA: ACM, 1995, pp. 163–174.

[4] E. Chavez, G. Navarro, R. A. Baeza-Yates, and J. L. Marroquin, "Searching in metric spaces," *ACM Computing Surveys*, vol. 33, no. 3, pp. 273–321, 2001.

[5] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," 2000.

[6] L. Sarmento, A. Kehlenbeck, E. C. Oliveira, and L. H. Ungar, "An approach to web-scale named-entity disambiguation," in *Machine Learning and Data Mining in Pattern Recognition, 6th International Conference, MLDM 2009, Leipzig, Germany, July 23-25, 2009. Proceedings*, ser. Lecture Notes in Computer Science, P. Perner, Ed., vol. 5632. Springer, 2009, pp. 689–703.

[7] A. Bagga and B. Baldwin, "Entity-based cross-document coreferencing using the vector space model," in *Proceedings of the 17th international conference on Computational Linguistics (ACL 1998)*, 1998, pp. 79–88.

[8] C. H. Gooi and J. Allan, "Cross-document coreference on a large scale corpus," in *HLT-NAACL'04*, 2004, pp. 9–16.

[9] M. Wick, S. Singh, and A. McCallum, "A discriminative hierarchical model for fast coreference at large scale," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ser. ACL '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 379–388.

[10] L. Kolb, A. Thor, and E. Rahm, "Load balancing for mapreduce-based entity resolution," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, 2012, pp. 618–629.

[11] A. Haghighi and D. Klein, "Unsupervised coreference resolution in a nonparametric bayesian model," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 848–855. [Online]. Available: http://www.aclweb.org/anthology/P07-1107

[12] J. H. Friedman and U. Fayyad, "On bias, variance, 0/1-loss, and the curse-of-dimensionality," *Data Mining and Knowledge Discovery*, vol. 1, pp. 55–77, 1997.