Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching

Andrew McCallum[‡†]
[‡]WhizBang! Labs - Research
4616 Henry Street
Pittsburgh, PA USA
mccallum@cs.cmu.edu

Kamal Nigam[†]
[†]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA USA
knigam@cs.cmu.edu

Lyle H. Ungar[*]
[*]Computer and Info. Science
University of Pennsylvania
Philadelphia, PA USA
ungar@cis.upenn.edu

ABSTRACT

Many important problems involve clustering large datasets. Although naive implementations of clustering are computationally expensive, there are established efficient techniques for clustering when the dataset has either (1) a limited number of clusters, (2) a low feature dimensionality, or (3) a small number of data points. However, there has been much less work on methods of efficiently clustering datasets that are large in all three ways at once—for example, having millions of data points that exist in many thousands of dimensions representing many thousands of clusters.

We present a new technique for clustering these large, high-dimensional datasets. The key idea involves using a cheap, approximate distance measure to efficiently divide the data into overlapping subsets we call *canopies*. Then clustering is performed by measuring exact distances only between points that occur in a common canopy. Using canopies, large clustering problems that were formerly impossible become practical. Under reasonable assumptions about the cheap distance metric, this reduction in computational cost comes without any loss in clustering accuracy. Canopies can be applied to many domains and used with a variety of clustering approaches, including Greedy Agglomerative Clustering, K-means and Expectation-Maximization. We present experimental results on grouping bibliographic citations from the reference sections of research papers. Here the canopy approach reduces computation time over a traditional clustering approach by more than an order of magnitude and decreases error in comparison to a previously used algorithm by 25%.

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*

1. INTRODUCTION

Unsupervised clustering techniques have been applied to many important problems. By clustering patient records, health care trends are discovered. By clustering address lists, duplicate entries are eliminated. By clustering astronomical data, new classes of stars are identified. By clustering documents, hierarchical organizations of information are derived. To address these applications, and many others, a variety of clustering algorithms have been developed.

However, traditional clustering algorithms become computationally expensive when the data set to be clustered is large. There are three different ways in which the data set can be large: (1) there can be a large number of elements in the data set, (2) each element can have many features, and (3) there can be many clusters to discover. Recent advances in clustering algorithms have addressed these efficiency issues, but only partially. For example, KD-trees [15] provide for efficient EM-style clustering of many elements, but require that the dimensionality of each element be small. Another algorithm [3] efficiently performs K-means clustering by finding good initial starting points, but is not efficient when the number of clusters is large. There has been almost no work on algorithms that work efficiently when the data set is large in all three senses at once—when there are millions of elements, many thousands of features, and many thousands of clusters.

This paper introduces a technique for clustering that is efficient when the problem is large in all of these three ways at once. The key idea is to perform clustering in two stages, first a rough and quick stage that divides the data into overlapping subsets we call “canopies,” then a more rigorous final stage in which expensive distance measurements are only made among points that occur in a common canopy. This differs from previous clustering methods in that it uses two different distance metrics for the two stages, and forms overlapping regions.

The first stage can make use of extremely inexpensive methods for finding data elements near the center of a region. Many proximity measurement methods, such as the inverted index commonly used in information retrieval systems, are very efficient in high dimensions and can find elements near the query by examining only a small fraction of a data set.

Variants of the inverted index can also work for real-valued data.

Once the canopies are built using the approximate distance measure, the second stage completes the clustering by running a standard clustering algorithm using a rigorous, and thus more expensive distance metric. However, significant computation is saved by eliminating all of the distance comparisons among points that do not fall within a common canopy. Unlike hard-partitioning schemes such as "blocking" [13] and KD-trees, the overall algorithm is tolerant to inaccuracies in the approximate distance measure used to create the canopies because the canopies may overlap with each other.

From a theoretical standpoint, if we can guarantee certain properties of the inexpensive distance metric, we can show that the original, accurate clustering solution can still be recovered with the canopies approach. In other words, if the inexpensive clustering does not exclude the solution for the expensive clustering, then we will not lose any clustering accuracy. In practice, we have actually found small accuracy *increases* due to the combined usage of two distance metrics.

Clustering based on canopies can be applied to many different underlying clustering algorithms, including Greedy Agglomerative Clustering, K-means, and Expectation-Maximization.

This paper presents experimental results in which we apply the canopies method with Greedy Agglomerative Clustering to the problem of clustering bibliographic citations from the reference sections of computer science research papers. The *Cora* research paper search engine [11] has gathered over a million such references referring to about a hundred thousand unique papers; each reference is a text segment existing in a vocabulary of about hundred thousand dimensions. Multiple citations to the same paper differ in many ways, particularly in the way author, editor and journal names are abbreviated, formatted and ordered. Our goal is to cluster these citations into sets that each refer to the same article. Measuring accuracy on a hand-clustered subset consisting of about 2000 references, we find that the canopies approach speeds the clustering by an order of magnitude while also providing a modest improvement in clustering accuracy. On the full data set we expect the speedup to be five orders of magnitude.

2. EFFICIENT CLUSTERING WITH CANOPIES

The key idea of the canopy algorithm is that one can greatly reduce the number of distance computations required for clustering by first cheaply partitioning the data into overlapping subsets, and then only measuring distances among pairs of data points that belong to a common subset.

The canopies technique thus uses two different sources of information to cluster items: a cheap and approximate similarity measure (e.g., for household address, the proportion of words in common between two address) and a more expensive and accurate similarity measure (e.g., detailed field-by-field string edit distance measured with tuned transformation costs and dynamic programming).

We divide the clustering process into two stages. In the first stage we use the cheap distance measure in order to create some number of overlapping subsets, called "canopies." A canopy is simply a subset of the elements (*i.e.* data points or items) that, according to the approximate similarity measure, are within some distance threshold from a central point. Significantly, an element may appear under more than one canopy, and every element must appear in at least one canopy. Canopies are created with the intention that points not appearing in any common canopy are far enough apart that they could not possibly be in the same cluster. Since the distance measure used to create canopies is approximate, there may not be a guarantee of this property, but by allowing canopies to overlap with each other, by choosing a large enough distance threshold, and by understanding the properties of the approximate distance measure, we can have a guarantee in some cases.

The circles with solid outlines in Figure 1 show an example of overlapping canopies that cover a data set. The method by which canopies such as these may be created is described in the next subsection.

In the second stage, we execute some traditional clustering algorithm, such as Greedy Agglomerative Clustering or K-means using the accurate distance measure, but with the restriction that we do not calculate the distance between two points that never appear in the same canopy, *i.e.* we assume their distance to be infinite. For example, if all items are trivially placed into a single canopy, then the second round of clustering degenerates to traditional, unconstrained clustering with an expensive distance metric. If, however, the canopies are not too large and do not overlap too much, then a large number of expensive distance measurements will be avoided, and the amount of computation required for clustering will be greatly reduced. Furthermore, if the constraints on the clustering imposed by the canopies still include the traditional clustering solution among the possibilities, then the canopies procedure may not lose any clustering accuracy, while still increasing computational efficiency significantly.

We can state more formally the conditions under which the canopies procedure will perfectly preserve the results of traditional clustering. If the underlying traditional clusterer is K-means, Expectation-Maximization or Greedy Agglomerative Clustering in which distance to a cluster is measured to the centroid of the cluster, then clustering accuracy will be preserved exactly when:

> For every traditional cluster, there exists a canopy such that all elements of the cluster are in the canopy.

If instead we perform Greedy Agglomerative Clustering clustering in which distance to a cluster is measure to the closest point in the cluster, then clustering accuracy will be preserved exactly when:

> For every cluster, there exists a set of canopies such that the elements of the cluster "connect" the canopies.

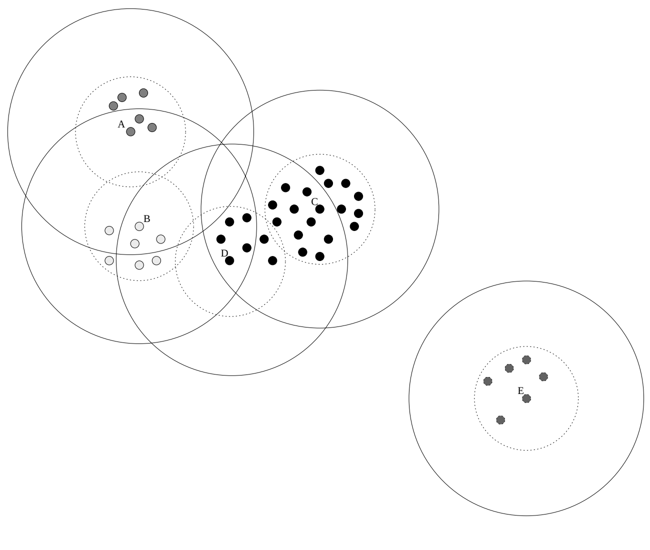

Figure 1: An example of four data clusters and the canopies that cover them. Points belonging to the same cluster are colored in the same shade of gray. The canopies were created by the method outlined in section 2.1. Point A was selected at random and forms a canopy consisting of all points within the outer (solid) threshold. Points inside the inner (dashed) threshold are excluded from being the center of, and forming new canopies. Canopies for B, C, D, and E were formed similarly to A. Note that the optimality condition holds: for each cluster there exists at least one canopy that completely contains that cluster. Note also that while there is some overlap, there are many points excluded by each canopy. Expensive distance measurements will only be made between pairs of points in the same canopies, far fewer than all possible pairs in the data set.

We have found in practice that it is not difficult to create inexpensive distance measures that nearly always satisfy these "canopy properties."

2.1 Creating Canopies

In most cases, a user of the canopies technique will be able to leverage domain-specific features in order to design a cheap distance metric and efficiently create canopies using the metric. For example, if the data consist of a large number of hospital patient records including diagnoses, treatments and payment histories, a cheap measure of similarity between the patients might be "1" if they have a diagnosis in common and "0" if they do not. In this case canopy creation is trivial: people with a common diagnosis fall in the same canopy. (More sophisticated versions could take account of the hierarchical structure of diagnoses such as ICD9 codes, or could also include secondary diagnoses.) Note, however, that people with multiple diagnoses will fall into multiple canopies and thus the canopies will overlap.

Often one or a small number of features suffice to build canopies, even if the items being clustered (e.g. the patients) have thousands of features. For example, bibliographic citations might be clustered with a cheap similarity metric which only looks at the last names of the authors and the year of publication, even though the whole text of the reference and the article are available.

At other times, especially when the individual features are noisy, one may still want to use all of the many features of an item. The following section describes a distance metric and method of creating canopies that often provides good performance in such cases. For concreteness, we consider the case in which documents are the items and words in the document are the features; the method is also broadly applicable to other problems with similar structure.

2.1.1 A Cheap Distance Metric

All the very fast distance metrics for text used by search engines are based on the inverted index. An inverted index is a sparse matrix representation in which, for each word, we can directly access the list of documents containing that word. When we want to find all documents close to a given query, we need not explicitly measure the distance to all documents in the collection, but need only examine the list of documents associated with each word in the query. The great majority of the documents, which have no words in common with the query, need never be considered. Thus we can use an inverted index to efficiently calculate a distance metric that is based on the number of words two documents have in common.

Given the above distance metric, one can create canopies as follows. Start with a list of the data points in any order, and with two distance thresholds, T_1 and T_2, where $T_1 > T_2$. (These thresholds can be set by the user, or, as in our experiments, selected by cross-validation.) Pick a point off the list and approximately measure its distance to all other points. (This is extremely cheap with an inverted index.) Put all points that are within distance threshold T_1 into a canopy. Remove from the list all points that are within distance threshold T_2. Repeat until the list is empty. Figure 1 shows some canopies that were created by this procedure.

The idea of an inverted index can also be applied to high-dimensional real-valued data. Each dimension would be discretized into some number of bins, each containing a balanced number of data points. Then each data point is effectively turned into a "document" containing "words" consisting of the unique bin identifiers for each dimension of the point. If one is worried about edge effects at the boundaries between bins, we can include in a data point's document the identifiers not only of the bin in which the point is found, but also the bins on either side. Then, as above, a cheap distance measure can be based on the the number of bin identifiers the two points have in common. A similar procedure has been used previously with success [8].

2.2 Canopies with Greedy Agglomerative Clustering

Greedy Agglomerative Clustering (GAC) is a common clustering technique used to group items together based on similarity. In standard greedy agglomerative clustering, we are given as input a set of items and a means of computing the distance (or similarity) between any of the pairs of items.

The items are then combined into clusters by successively combining the two closest clusters until one has reduced the number of clusters to a target number.

We use a standard implementation of greedy agglomerative clustering (GAC): Initialize each element to be a cluster of size one, compute the distances between all pairs of clusters, sort the distances from smallest to largest, and then repeatedly merge the two clusters which are closest together until one is left with the desired number of clusters.

In the standard GAC implementation, we need to apply the distance function $O(n^2)$ times to calculate all pair-wise distances between items. A canopies-based implementation of GAC can drastically reduce this required number of comparisons.

Using a cheap, approximate distance measure overlapping canopies are created. When the canopies property holds, we are guaranteed that any two points that do not share a canopy will not fall into the same cluster. Thus, we do not need to calculate the distances between these pairs of points. Equivalently, we can initialize all distances to infinity and only replace pairwise distances when two items fall into the same canopy. As discussed in Section 2.4, this vastly reduces the required number of distance calculations for greedy agglomerative clustering.

2.3 Canopies with Expectation-Maximization Clustering

One can also use the canopies idea to speed up prototype-based clustering methods like K-means and Expectation-Maximization (EM). In general, neither K-means nor EM specify how many clusters to use. The canopies technique does not help this choice.

As before, the canopies reduce the number of expensive distance comparisons that need to be performed. We create the canopies as before. We now describe three different methods of using canopies with prototype-based clustering techniques.

Method 1: In this approach, prototypes (our estimates of the cluster centroids) are associated with the canopies that contain them, and the prototypes are only influenced by data that are inside their associated canopies.

After creating the canopies, we decide how many prototypes will be created for each canopy. This could be done, for example, using the number of data points in a canopy and AIC or BIC [1]—where points that occur in more than one canopy are counted fractionally. Then we place prototypes into each canopy. This initial placement can be random, as long as it is within the canopy in question, as determined by the inexpensive distance metric.

Then, instead of calculating the distance from each prototype to every point (as is traditional, a $O(nk)$ operation), the E-step instead calculates the distance from each prototype to a much smaller number of points. For each prototype, we find the canopies that contain it (using the cheap distance metric), and only calculate distances (using the expensive distance metric) from that prototype to points within those canopies.

	Prototypes moved by	**Number of prototypes**
1	only points in same canopy as prototype	constant, initialized per canopy
2	points in same canopy as prototype plus all others summarized by canopy centers	constant, initialized over whole data set
3	only points in same canopy as prototype	initialized per canopy, but created and destroyed dynamically

Table 1: A summary of the three different methods of combining canopies and EM.

Note that by this procedure prototypes may move across canopy boundaries when canopies overlap. Prototypes may move to cover the data in the overlapping region, and then move entirely into another canopy in order to cover data there.

The canopy-modified EM algorithm behaves very similarly to traditional EM, with the slight difference that points outside the canopy have no influence on points in the canopy, rather than a minute influence. If the canopy property holds, and points in the same cluster fall in the same canopy, then the canopy-modified EM will almost always converge to the same maximum in likelihood as the traditional EM. In fact, the difference in each iterative step (apart from the enormous computational savings of computing fewer terms) will be negligible since points outside the canopy will have exponentially small influence.

K-means gives not just similar results for canopies and the traditional setting, but exactly identical clusters. In K-means each data point is assigned to a single prototype. As long as the cheap and expensive distance metrics are sufficiently similar that the nearest prototype (as calculated by the expensive distance metric) is within the boundaries of the canopies that contain that data point (as calculated with the cheap distance metric), then the same prototype will always win.

Method 2: We might like to have a version of the canopies method that, instead of forcing us to pick a number of prototypes for each canopy separately, allows us to select the number of prototypes that will cover the whole data set. Method 2 makes this possible. As before, prototypes are associated with canopies, and are only influenced by individual data points that are inside their canopies. However, in this method, prototypes are influenced by all the other data too, but data points outside the prototype's associated canopy are represented simply by the mean of their canopies. In this respect, Method 2 is identical to Omohundro's balltrees [17]. Method 2 differs, however, in that the canopies are created highly efficiently, using a cheap distance metric. Not only is it more computationally efficient to compute the distance between two points using the cheap distance metric, but the use of inverted indices avoids completely computing the distance to many points.

Method 3: There are many existing traditional methods for dynamically determining the number of prototypes (e.g. [18]). Techniques for creating and destroying prototypes are particularly attractive when thinking about Method 1. Here we have the simplicity of completely ignoring all data points outside a prototype's associated cluster, with the problem, however, that we may have too many or too few prototypes within the canopy. In Method 3, as in Method 1, prototypes are associated with canopies and only see points within their canopy. Here, however, we use techniques that create (and possibly destroy) prototypes dynamically during clustering. We avoid creating multiple prototypes to cover points that fall into more than one canopy by invoking a "conservation of mass" principle for points. In practice, this means that the contribution of each point is divided among the canopies, as falling out naturally from the normalization in the E-step: as is traditional, membership to a prototype is determined by dividing the inverse distance to the prototype by the sum of inverse distances to *all the prototypes to which its distance was measured*, even if some of those prototypes fall in different canopies.

2.4 Computational Complexity

We can formally quantify the computational savings of the canopies technique. The technique has two components: a relatively fast step where the canopies are created, followed by a relatively slow clustering process. If we create canopies using an inverted index, we do not need to perform even the complete pair-wise distance comparisons. If we assume that each document has w words, and these words are evenly distributed across the vocabulary V, then we can compare a document to n other documents in $O(nw^2/|V|)$. This canopy creation cost is typically insignificant in comparison to the more detailed clustering phase.

Assume that we have n data points that originated from k clusters. For concreteness, first consider the Greedy Agglomerative Clustering algorithm. Clustering without canopies requires calculating the distance between all pairs of points, an $O(n^2)$ operation. This is the dominating complexity term in GAC clustering. Now consider how this is reduced by using canopies. Assume that there are c canopies and that each data point on average falls into f canopies. This factor f estimates the amount to which the canopies overlap with each other. Now, in an optimistic scenario where each canopy is of equal size, there will be roughly fn/c data points per canopy. When clustering with canopies we need only calculate the distances between pairs of points within the same canopy. This requires at most $O(c(fn/c)^2) = O(f^2n^2/c)$ distance measurements. (This is probably an over-estimate, as the same points tend to fall into multiple canopies together, and we only need to calculate their distances once.) This represents a reduction in complexity of f^2/c. In general, c will be much larger than f. Given a typical case in which $n = 1,000,000$, $k = 10,000$, $c = 1,000$, and f is a small constant, the canopies technique reduces computation by a factor of $1,000$.

In the case of K-means or Expectation-Maximization, clustering without canopies requires $O(nk)$ distance comparisons per iteration of clustering (finding the distance between each data point and each cluster prototype). Consider the EM method 1, where each cluster belongs to one or more canopy. Assume that clusters have roughly the same overlap factor f as data points do. Then, each cluster needs to compare itself to the fn/c points in f different canopies. For all clusters, this is $O(nkf^2/c)$ per iteration, yielding the same complexity reduction as seen for GAC.

In the experimental results described in a following section, c is on the order of 1,000, so the savings are significant. In an industrial sized merge-purge problem, far more canopies would be used, and full pair-wise distance calculations would not at all be feasible.

Fahlman, Scott & Lebiere, Christian (1989). The cascade-correlation learning architecture. In Touretzky, D., editor, Advances in Neural Information Processing Systems (volume 2), (pp. 524-532), San Mateo, CA. Morgan Kaufmann.

Fahlman, S.E. and Lebiere, C., "The Cascade Correlation Learning Architecture," NIPS, Vol. 2, pp. 524-532, Morgan Kaufmann, 1990.

Fahlmann, S. E. and Lebiere, C. (1989). The cascade-correlation learning architecture. In Advances in Neural Information Processing Systems 2 (NIPS-2), Denver, Colorado.

Figure 2: Three sample citations to the same paper. Note the different layout formats, and the mistakes in spelling that make it difficult to identify these as citations to the same paper.

3. CLUSTERING TEXTUAL BIBLIOGRAPHIC REFERENCES

In this section we provide empirical evidence that using canopies for clustering can increase computational efficiency by an order of magnitude without losing any clustering accuracy. We demonstrate these results in the domain of bibliographic references.

The Cora web site (*www.cora.whizbang.com*) provides a search interface to over 50,000 computer science research papers [11]. As part of the site's functionality, we provide an interface for traversing the citation graph. That is, for a given paper, we provide links to all other papers it references, and links to all other papers that reference it, in the respective bibliography section of each paper. To provide this interface to the data, it is necessary to recognize when two citations from different papers are referencing the same third paper, even though the text of the citations may differ. For example, one paper may abbreviate first author names, while the second may include them. Some typical citations are shown in Figure 2. Note that different styles of reference formatting and abbreviation, as well as outright citation mistakes, make this a difficult task to automate. We pose this as a clustering problem, where the task is to take all the citations from all papers in the collection, and cluster them so that each cluster contains all and only citations to a single paper. Since the Cora collection contains over a million bibliographic entries, it is necessary to perform this clustering efficiently. Using straightforward GAC would take more than one CPU year, assuming unlimited memory. If we estimate the total number of canopies and average canopy membership from labeled dataset used below, the canopies approach will provide a speedup of five orders of magnitude, reducing the clustering time to a couple hours.

3.1 Distance Metrics for Citations

The basic clustering approach we use is Greedy Agglomerative Clustering. In order to perform clustering in this domain, we must provide a distance metric for the space of bibliographic citations. A powerful choice for measuring the distance between strings is string edit distance, as calculated by dynamic programming using different costs associated with various transformation rules [21].

Since we know that that the strings are references, we choose transformation cost parameters specific to this domain. There are different transformation costs for (1) deleting a character, (2) deleting a character where the last operation was also a deletion, (3) deleting a period, (4) deleting a character in one string when the other string is currently at a period, (5) substituting one character for another character, (6) substituting a non-alphabetic character for another non-alphabetic character, and (7) deleting a non-alphabetic character.

One difficulty with applying string edit distances to the domain of citations is that one cannot easily represent field transpositions (e.g. placing the year after the author instead of at the end of the citation) as an atomic cost operation in the dynamic programming. We expect there to be strong similarity between the individual fields of citations to the same paper, but do not expect strong correlations in the ordering of the fields. Thus, we define our distance metric to be the weighted average of the distances of each of the fields occurring in the citations. The fields of each citation are found automatically using a hidden Markov model [11]; this field extraction process is not described in this paper.

The string edit distance calculations are relatively expensive, since a dynamic program must be solved to account for possible insertions, deletions, and transpositions. Doing a full clustering for the Cora dataset would involve solving over one trillion of these dynamic programs, and is clearly untenable. To make this feasible, we use the canopies approach with a two-pass process, first using an inexpensive distance metric to limit the number of string edit distances we must compute.

As described in section 2.1.1, we use an inverted index and the method of two thresholds to inexpensively create the canopies. Specifically, the distance between two citations is measured by considering each citation as a short document, and measuring similarity using the text search engine *Archer* [12]. Similarity between two citations is the standard TFIDF cosine-similarity from information retrieval [19]. This coarse metric allows approximate overlapping canopies to be created very quickly.

3.2 Dataset, Protocol and Evaluation Metrics

In order to evaluate both the speed and accuracy of our clustering algorithms, we take a subset of the Cora citation data and hand-label it according to its correct clustering. Our labeled data consist of all citations from our collection to papers authored by either Michael Kearns, Robert Schapire or Yoav Freund. We identify these papers by generous substring matching on the author names, and then prune out papers not authored by one of our subjects. In all, this comprises 1916 citations to 121 distinct papers. About one-quarter of the papers have only one or two reference to them; several papers have many references to them. The most popular paper is cited 108 times.

To cluster the citations we use the two-thresholds canopy creation and then nearest-neighbor Greedy Agglomerative Clustering with the string edit distances on the extracted fields as the expensive distance metric. We use only four fields of the citation for the string edit distance calculation: author, title, date, and venue (e.g. journal name or conference name). All fields are lower-cased. Dates are reduced to a four-digit year. Titles and venues are truncated at 60 characters. Author fields are simplified by automatically abbreviating the first name of the first author. The HMM makes a number of errors, which build on the natural variation in citations. The worst of these are fixed in a pre-processing stage.

There are three tunable parameters for the canopies clustering: the tight and loose thresholds for the cheap distance metric, and the stopping point for the Greedy Agglomerative Clustering. We tuned these three parameters on a separate, similarly sized validation dataset for the authors Scott Fahlman, Dennis Kibler and Paul Utgoff. The string edit costs for the different operations were set to hand-coded values. Learning and tuning these string edit weights automatically is an area of ongoing research.

In analyzing the results, we present several metrics for evaluating the quality of the clustering. All our metrics are defined in terms of all pairs of citations. Thus, the error rate is the fraction of pairs of citations that are correctly in the same cluster (if they reference the same paper) or in different clusters (if they reference different papers). Since the vast majority of pairs of citations fall into different clusters, error is not the most informative metric. Thus, we consider the precision, recall and the F1 metric that do not credit an algorithm for correctly placing citations into different clusters. These metrics are standard measures used in information retrieval. Precision is the fraction of correct predictions among all pairs of citations *predicted* to fall in the same cluster. Recall is the fraction of correct predictions among all pairs of citations that *truly* fall in the same cluster. F1 is the harmonic average of precision and recall. It is a single summary statistic that does not credit an algorithm for correctly placing the overwhelming number of pairs into different clusters.

We use a series of four increasingly sophisticated methods for comparing the computational efficiency and clustering accuracy of using the canopy framework. The most naive baseline is to simply place each citation into its own cluster. This represents the straw-man performance if no computation is used for the clustering. As a second baseline, we create by hand a regular expression that identifies the last name of the first author, and the year of each citation. Each citation is placed into the same cluster as any other paper published in the same year by an author with the same last name. This baseline represents a bibliographic map done with a large amount of manual tuning, but without an automatic clustering algorithm. As a third comparison, we use the current grouping algorithm used by Cora [11]. This is a word-matching algorithm that is similar to the two thresh-

Method	F1	Error	Precision	Recall	Minutes
Canopies	0.838	0.75%	0.735	0.976	7.65
Complete, Expensive	0.835	0.76%	0.737	0.965	134.09
Existing Cora	0.784	1.03%	0.673	0.939	0.03
Author/Year Baseline	0.697	1.60%	0.559	0.926	0.03
Naive Baseline	—	1.99%	1.000	0.000	—

Table 2: The error and time costs of different methods of clustering references. The naive baseline places each citation in its own cluster. The Author/Year baseline clusters each citation based on the year and first author of the citation, identified by hand-built regular expressions. The existing Cora method is a word matching based technique. Note that the canopy clustering is much quicker, and slightly more accurate than the complete clustering. Time is measured in minutes for the Perl implementations (except the existing Cora, which is implemented in C).

		Loose Threshold				
		0.95	0.85	0.75	0.65	0.55
	0.95	0.566 (7.15)	0.667 (9.13)	0.835 (10.72)	0.836 (18.77)	0.836 (35.53)
Tight	0.85	—	0.713 (5.85)	0.833 (8.12)	0.835 (10.67)	0.836 (24.10)
Threshold	0.75	—	—	0.840 (5.50)	0.836 (8.80)	0.836 (24.17)
	0.65	—	—	—	**0.838 (7.65)**	0.836 (15.97)
	0.55	—	—	—	—	0.836 (15.93)

Table 3: F1 results created by varying the parameters for the tight and loose thresholds during canopy creation. The number in parenthesis is the computation time in minutes for the clustering. As we decrease the loose threshold and increase the tight threshold, computation time increases as we are required to calculate more pairwise expensive distances. The parameter setting chosen by our cross-validation is indicated in bold.

olds matching, except that (1) it uses only the titles, authors and years from each citation, and (2) it creates a clustering that is non-overlapping. Our final method is to perform the complete hierarchical agglomerative clustering with the string edit distance metric. This represents a very expensive baseline, but one that should perform accurately. All methods are implemented in Perl, except for the existing Cora algorithm, which is implemented in C. Experiments are run on a 300 MHz Pentium-II with enough memory that there is no paging activity.

3.3 Experimental Results

Table 3.2 presents a summary of the experimental results with the canopy clustering, where the threshold parameters and the number of clusters are tuned on the separate validation set. The canopy clustering algorithm achieves an F1 of 0.838 in only 7.65 minutes. In comparison, the complete clustering takes over two hours, and has slightly worse error. Note that this represents more than an order of magnitude reduction in computation time.

Although the original motivation for the algorithm was computation time and size, the canopies approach also provides a slightly more accurate clustering algorithm. Our hypothesis about this somewhat surprising result is that the two levels of clustering (the loose canopies and the strict string edit) work together in some sense. The cheap distance metric produces errors mostly independent from those of the expensive distance metric. This allows the cheap canopy partitioning to remove some errors that the expensive clustering would have made.

The other comparison techniques all give worse clustering solutions than those using canopies. The error of the baseline naive approach is more than twice that of either clustering approach. Using either the author/year or the existing Cora techniques improves upon the baseline accuracy, but is still significantly worse than either clustering technique.

In performing a canopy clustering, two sets of parameters need to be chosen: the values for the two canopy thresholds, and the number of final clusters. Table 3 shows experimental results that vary the values of the tight and the loose thresholds. The number of comparisons done by the expensive clustering algorithms varies when the thresholds change, because the number and size of the canopies changes. In general as the tight similarity threshold increases and the loose threshold decreases, more distance measurements are required. For example, with {Tight=0.95, Loose=0.55} 261,072 expensive distance calculations are required. With {Tight=0.75, Loose=0.75}, the best performance seen, only 41,142 distances are required, nearly six times less. This means that the canopy creation is finding pairs that belong together, even if their distances are not so close. As a comparison, doing the complete clustering, without canopies, requires 1,834,570 expensive distance calculations, and the parameters picked by the validation set, {Tight=0.65, Loose=0.65}, required 41,141. With larger datasets, this difference becomes even more pronounced, as the number of distance calculations required by the full clustering grows by the square of the number of items.

Table 4 shows how the error of the canopy clustering varies when the final number of clusters is changed. Note that having the correct number of clusters (121), or slightly more, provides the best accuracy. Detailed error analysis shows that most of our error (85% of it) comes from citations that

# Clusters	Distance	F1	Precision	Recall
260	4	0.789	0.809	0.770
189	6	0.807	0.752	0.871
143	8	0.833	0.746	0.942
129	10	0.837	0.742	0.960
121	12	0.839	0.742	0.965
110	14	0.838	0.735	0.975
105	16	0.812	0.694	0.980
100	18	0.791	0.663	0.981
95	20	0.756	0.614	0.983
91	22	0.752	0.609	0.984
90	24	0.752	0.609	0.984

Table 4: The accuracy of the clustering as we vary the final number of clusters. Note that the best F1 score occurs near the correct number of clusters (121). As we allow more clusters, the precision increases, while the recall decreases.

should be in different clusters, but are predicted to be in the same cluster. Canopy clustering still make three times as many mistakes falsely putting references in the same cluster, rather than falsely putting references in different clusters.

4. RELATED WORK

Many researchers and practitioners have recognized the desirability—and the difficulty—of grouping similar items in large data sets into clustering. The methods used tend to fall into two categories: database methods for finding near duplicate records and clustering algorithms.

The extensive literature on clustering algorithms goes back many years. (See e.g. [2].) Almost all of the methods use some single method of computing similarity between pairs of items. These items are then either greedily or iteratively clustered based on their similarity. Standard clustering methods include: greedy agglomerative methods, greedy divisive methods, and iterative methods such as K-means clustering. More recent variants on these classical clustering methods use (iterative) EM methods to estimate parameters in formal statistical models or use other forms of overlapping clusters to improve precision [22]. They may also represent subsets of the data by single points [23] or use incremental clustering to improve clustering speed on large data sets [20]. The EM and statistical methods tend to be slow, while the one-pass incremental methods tend to be less accurate.

The canopies method described here differs from the above methods in that is makes use of two different similarity measures. By using the cruder similarity measure to quickly form canopies and then using the more refined similarity measure to form smaller clusters, both high speed and high precision are obtained.

Closely related to the above methods are a large number of extensions to and variants on KD-trees [5] such as multi-resolution KD-trees [15], which recursively partition the data into subgroups. Almost all of these methods suffer from doing hard partitions, where each item must be on a single side of each partition. Cheap, approximate similarity measures thus cannot be used, since if an item is put on the wrong side of a partition there is no way to later correct the error. KD-tree methods also typically scale poorly for items with large numbers of attributes, as splits are generally made on a single attribute. The *balltrees* method [17] does use overlapping regions, but still assumes that a tree structure can be made. Like the other KD-tree methods, it does not make use of the two (expensive and cheap) similarity measures upon which the canopies method is based.

A separate line of work on clustering comes from users of databases. The *record linkage* [16, 4, 10] or *merge–purge* [7] problem occurs when a company purchases multiple databases, such as multiple mailing lists, and wishes to determine which records are near duplicates (i.e. refer to the same person) so that the lists can be merged and the duplicates purged. This is a special case of clustering problem addressed above, in which the clusters are small and items in each cluster are typically quite distinct from items in other clusters.

In one approach to the merge–purge problem, multiple different keys are used to determine "duplicate" records, and the results of those different clusters are combined [7]. For example, a database is created by combining multiple databases which may contain duplicate or near duplicate records. This composite database is sorted separately on multiple keys (address, social security number, etc.). For each sorting, items within a small window of each other are checked to see if they are near duplicates, and if so combined. In a closely related approach, the database is sorted using an application-specific key (e.g. last name) and then a more expensive similarity is computed between items that are close in the sorted list [13, 14]. Like the other merge–purge researchers, Monge and Elkan assume that an exact match in a single field establishes whether two items are duplicates; our canopies algorithm allows for more complex comparisons, such as are used in clustering methods like K-means, and is more tolerant of noise. Explicitly assigning items to multiple canopies allows rigorous methods such as EM to be used within each canopy, and clear computational cost estimates to be made.

Hylton [9] comes even closer to our canopies algorithm. He proposes a method of clustering references to published articles which uses a two step algorithm. In the first step, for each item (i.e. for each reference) a pool of potentially matching items is selected by doing three full-text searches, where the query for each search consists of one of the authors' last names and two words from the title, selected at random. In the second step, all items in each pool are compared against the item used to generate the pool, using a string matching algorithm. He then forces all items which have been grouped together into a single group. For example, if items A and B were grouped together based on the queries about A and items B and C were grouped together in the queries about C, then A, B, and C would end up in the same group.

Hylton's method follows the spirit of canopies in that it does an initial rough grouping which is not a unique partition, followed by a more fine-grained comparison of items within each group. It differs from canopies in that he forms one group ("pool", in his terms) for every single item. (In our terms, he requires the number of canopies to equal the number of items.) This is very expensive. Also, because each pool is centered on a single item, Hylton does not support the use of arbitrary clustering methods for the fine-grained clustering portion of the algorithm.

Giles et al. [6] also study the domain of clustering citations. They present experiments with several different word-matching techniques and one string edit based technique. They find that the best-performing method is a word matching algorithm similar to the Cora technique used in Section 3, augmented to use field extraction information and bigrams. Their lesser-performing string edit distance metric is similar to ours, but the clustering algorithm they use with this metric is not greedy agglomerative clustering. Instead, they use an iterative match-and-remove algorithm to perform the clustering.

5. CONCLUSIONS

Clustering large data sets is a ubiquitous task. Astronomers work to classify stars into similar sets based on their images. Search engines on the web seek to group together similar documents based on the words they contain or based on their citations. Marketers seek clusters of similar shoppers based on their purchase history and demographics. Shopbots seek to identify similar products based on the product descriptions. Biologists seek to group DNA sequences based on the proteins they code for or to group proteins based on their function in cells.

In all of these cases, the objects are characterized by large feature vectors (the images, text, or DNA sequences). Furthermore, in each case there are inexpensive measures of similarity that one can compute (e.g. number of words in common), and more expensive and accurate measures (e.g. based on parse trees and part of speech tagging for text or string-edit distances for DNA). Given large data sets with hundreds of thousands or millions of entries, computing all pairwise similarities between objects is often intractable, and more efficient methods are called for. Also, increasingly, people are trying to fit complex models such as mixture distributions or HMMs to these large data sets. Computing global models where all observations can affect all parameters is again intractable, and methods for grouping observations (similarly to the grouping of objects above) are needed. Canopies provide a principled approach to all these problems.

In this paper we have focused on *reference matching*, a particular class of problems that arise when one has many different descriptions for each of many different objects, and wishes to know (1) which descriptions refer to the same object, and (2) what the best description of that object is. We present experimental results for the domain of bibliographic citation matching. Another important instance of this class is the *merge-purge problem*. Companies often purchase and merge multiple mailing lists. The resulting list then has multiple entries for each household. Even for a single person, the name and address in each version on the list may differ slightly, with middle initials present or absent, words abbreviated or expanded, zip codes present or absent. This problem of merging large mailing lists and eliminating duplicates, becomes even more complex for *householding*, where one wishes to collapse the records of multiple people who live in the same household.

When information is extracted from the web, the reference matching problem is even more severe. One can search the web and extract descriptions of products and their attributes (e.g. different models of Palm Pilots and their weight, memory, size, etc.) or descriptions of companies and what industries and countries in which they have business. This information extraction is error-prone, but redundant—many different sources sell the same product. Again, the goal is to cluster descriptions into sets that describe the same product or company, and then to determine a canonical description.

Because of the the large number of items, feature dimensions and clusters, carefully comparing every item against every other item is intractable in all the above cases. Fortunately, there are often cheap and approximate means to group items into overlapping subsets we call "canopies", so that accurate, expensive comparisons can be made between items in the same canopy. Canopies, ideally, have the property that all items in any true cluster fall in the same canopy; this guarantees that no accuracy is lost by restricting comparisons of items to those in the same canopy.

The canopy approach is widely applicable. The cheap measures can be binning, comparison of a few attributes of a complex record, or finding similarity using an inverted index. The expensive measure can use detailed similarity measurements such as string edit distance computed with dynamic programming. The clustering can be greedy agglomerative, K-nearest neighbor, K-means, or any of a wide variety of EM methods. The commonality across the methods is the creation of canopies using the cheap measure so that the use of the expensive measure is drastically reduced.

We have demonstrated the success of the canopies approach on a reference matching problem from the domain of bibliographic citations. Here we reduced computation time by more than an order of magnitude while also slightly increasing accuracy. In ongoing work we are running the canopies algorithm on the full set of over a million citations and expect reduced computation of five orders of magnitude, from more than one CPU year to a couple hours. In future work we will quantify analytically the correspondence between the cheap and expensive distance metrics, and we will perform experiments with EM and with a wider variety of domains, including data sets with real-valued attributes.

Acknowledgments

Most of this work was done while the first two authors were at Just Research (www.justresearch.com). Although Cora is now supported by WhizBang! Labs, it was originally created at Just Research.

6. REFERENCES

[1] H. Akaike. On entropy maximization principle. *Applications of Statistics*, pages 27–41, 1977.

[2] M. R. Anderberg. *Cluster Analysis for Application.* Academic Press, 1973.

[3] P. S. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proc. 4th*

International Conf. on Knowledge Discovery and Data Mining (KDD-98). AAAI Press, August 1998.

[4] I. P. Felligi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Society*, 64:1183–1210, 1969.

[5] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Tras. Math. Software*, 3(3):209–226, 1977.

[6] C. L. Giles, K. D. Bollacker, and S. Lawrence. CiteSeer: An automatic citation indexing system. In *Digital Libraries 98 – Third ACM Conference on Digital Libraries*, 1998.

[7] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *Proceedings of the 1995 ACM SIGMOD*, May 1995.

[8] H. Hirsh. Integrating mulitple sources of information in text classification using whril. In *Snowbird Learning Conference*, April 2000.

[9] J. Hylton. Identifying and merging related bibliographic records. MIT LCS Masters Thesis, 1996.

[10] B. Kilss and W. Alvey, editors. *Record Linkage Techniques—1985*, 1985. Statistics of Income Division, Internal Revenue Service Publication 1299-2-96. Available from http://www.fcsm.gov/.

[11] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 2000. To appear.

[12] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/∼mccallum/bow, 1996.

[13] A. Monge and C. Elkan. The field-matching problem: algorithm and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, August 1996.

[14] A. Monge and C. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *The proceedings of the SIGMOD 1997 workshop on data mining and knowledge discovery*, May 1997.

[15] A. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In *Advances in Neural Information Processing Systems 11*, 1999.

[16] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[17] S. Omohundro. Five balltree construction algorithms. Technical report 89-063, International Computer Science Institute, Berkeley, California, 1989.

[18] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998.

[19] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[20] M. Sankaran, S. Suresh, M. Wong, and D. Nesamoney. Method for incremental aggregation of dynamically increasing database data sets. *U.S. Patent 5,794,246*, 1998.

[21] D. Sankoff and J. B. Kruskal. *Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley, 1983.

[22] J. W. Tukey and J. O. Pedersen. Method and apparatus for information access employing overlapping clusters. *U.S. Patent 5,787,422*, 1998.

[23] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, 1996.