Real-Time User-Centric Management of Time-Intensive Analytics Using Convergence of Local Functions

Invited position paper

Vinay Deolalikar
HP-Autonomy Research
vinay.deolalikar@hp.com

Abstract

The past decade has witnessed an astonishing growth in unstructured information in enterprises. The commercial value locked in enterprise unstructured information is being increasingly recognized. Accordingly, a range of textual document analytics—clustering, classification, taxonomy generation, provenance, etc.— have taken center stage as a potential means to manage this explosive growth in unstructured enterprise information, and unlock its value.

Several analytics are time-intensive: the time taken to complete processing the increasingly large volumes of data is significantly more than real-time. However, users are increasingly demanding *real-time* services that rely on such time-intensive analytics. There is clearly a tension between the aforementioned two developments.

In light of the preceding, vendors increasingly realize that *while an analytic may take a longer time to converge, they need to extract useful information from it in real-time. Furthermore, this information has to be application-driven.* In other words, it is often not an option to simply "wait until the analytic has finished running:" they must start providing the user with information while the analytic is still running. In summary, there is an emerging stress in Enterprise Information Management (EIM) on application-driven real-time information being extracted from time-intensive analytics.

A priori, it is not clear what could be extracted from an analytic that has yet to complete, and whether any such information would be useful. As of the present, there is little or no research literature on this problem: it is generally assumed that all of the information from an analytic will be available upon its completion.

We present an approach to this problem that is based on decomposing the objective function of the analytic, which is a global function that determines the progress of the analytic, into *multiple local, user-centric functions.* How can we construct meaningful local functions? How can such functions be measured? How do these functions evolve with time? Do these functions encode useful information that can be obtained real-time? These are the questions we will address in this paper.

We demonstrate our approach using local functions on document clustering using the de facto standard algorithm—*k*-means. In this case, the multiple local user-centric functions transform *k*-means into a flow algorithm, with each local function measuring a flow. Our results show that these flows evolve very differently from the global objective function, and in particular, may often converge quickly at many local sites. Using this property, we are able to extract useful information considerably earlier than the time taken by *k*-means to converge to its final state.

We believe that such pragmatic approaches will have to be taken in order to manage systems performing analytics on large volumes of unstructured data.

1 Introduction

1.1 Enterprise Information Management

Enterprises spend billions of dollars annually to manage *unstructured information*; namely information that exists mostly as text in documents having multiple formats, but no fixed schema (unlike, say, a database which is queried using SQL). These documents reside on desktops, laptops, email exchanges, web and file servers, wikis, and sharepoint repositories. This segment of enterprise information is growing much faster than structured information, and already it is estimated that 70% of all information in an enterprise exists in unstructured formats.

Due to the lack of structure, managing unstructured information poses unique challenges. Currently, major drivers for these management efforts include applications such as eDiscovery, compliance requirements for different categories of documents necessitated by new laws such as HIPAA, IT management operations, document searches made by employees in various capacities, sales force support needs, and a host of other applica-

tions. Analytics[1] of various types—clustering, classification, taxonomy extraction, to name a few prominent ones—are generally regarded as the primary techniques that will help meet these challenges and enable such applications. Analytics for enterprise unstructured information, viewed through the prism of end-user applications, form the broad context for our work.

1.2 Emerging Problem: Real-Time Extraction of Information

Analytics, generally speaking, uncover relationships in unstructured information. The greater the volume of the information, the more time it takes for an analytic to process the information, and extract relationships. Workflows in enterprise information management are increasingly complex, and require analytics inputs at various stages. These stages are pipelined together. Users of these applications demand the ability to perform workflows in near real-time: any application that requires the user to "wait until the current stage completes" is a significant dent on market acceptance.

Therefore, on the one hand, analytics are needed to enable applications that require unstructured information management at scale. On the other hand, the time taken by a particular analytic to complete at scale prevents its use in the application.

In light of the above catch-22 situation, there has been a great deal of attention devoted to making analytics run faster. We wish, instead, to highlight an emerging paradigm: vendors are increasingly trying to obtain "just enough" information from an analytic that can satisfy the current need of the user, and enable them to move to the next stage of their workflow. In other words, they want to provide enough information to the user that hides the actual run time of the analytic from them. The analytic may well take significantly longer to complete, but how can we extract useful application-enabling information from it in near real-time? As of now, there is little or no work on this question, and to our knowledge we are the first to frame it explicitly. We believe that this question will become increasingly important as the scales of unstructured information grow.

1.3 Our Approach: Local User-Centric Functions

Most analytics try to optimize (usually minimize or maximize) some global *objective function*. For example, clustering tries to minimize the sum of distances of documents to cluster centroids. However, these objective functions are mathematical objects: end-users do not usually think in terms of objective functions. Rather, they have more application-centric concerns.

Our approach is to try to "partition" the global objective function into local functions that are user and application-centric, and that capture what the user might be interested in from the analytic. Then, we will try to measure these local functions. The hope is that while the global objective function captures the overall convergence behavior of the analytic, these local functions might already start yielding information to the user that is precisely of the form that they are interested in. The idea is that while the global state of the analytic is determined by the global objective function, the evolution of local states might be tracked using our local functions. These local functions, if they are appropriately constructed, might give users information that they can start acting upon immediately. Furthermore, by viewing the analytic as a conglomeration of locally evolving states, we can provide information to the user "piecemeal" instead of all at once, especially since that more accurately reflects how the user will digest the information anyway.

1.4 Contributions

Our main contributions are sketched below:

1. We frame the question of real-time piece-by-piece extraction of information from time-intensive analytics. We believe that this question will be increasingly important in the future, given the explosive growth of unstructured information.
2. We present a novel approach to the problem above, based on inspecting analytics algorithms locally using user-defined functions.
3. We work out our approach for an important analytic—text clustering—that is key to several EIM applications.

2 Key Idea: Defining User-Centric Objectives with Local Functions

As stated in the introduction, most analytics are defined in terms of an objective function that is to be maximized/minimized over the course of the run-time of the analytic. For example, k-means clustering is often defined as follows. Given m data-points $x = \{x_1, \ldots, x_m\}$, each of which is a d-dimensional vector, find k "means" $\mu = \{\mu_1, \ldots, \mu_k\}$, also d-dimensional, such that the following objective function is minimized.

$$E(x, \mu) = \sum_{i=1}^{m} L(x_i, \mu_j), \tag{1}$$

where μ_j is the closest of the k means to x_i in terms of the norm L.

[1] An analytic is, broadly, a functionality that examines data, analyzes it, and draws inference based upon the results of the analysis.

We may write the function above as a sum of sums: each outer sum would pertain to a single μ_j. Thus,

$$E(x,\mu) = \sum_{i=1}^{m} \sum_{j=1}^{k} L(x_i,\mu_j), \qquad (2)$$

where the inner sum is over all data-points that are closest to μ_j. The general form of the objective function then becomes "optimize some global function (the sum, in the case above) of local pieces." A "local piece" here is the inner sum that pertains only to a single cluster, and therefore *can be computed locally at each cluster*.

Moreover, as noted earlier, objective functions such as (1) are far from the mind of the end-user of an analytic. The user is concerned with something that *describes the problem from their perspective*. The enterprise user frequently wants to associate some meaning to the information that the analytic extracts as it runs.

Can we then, partition the objective function into local pieces, with the additional desideratum of making each local piece pertain to the user's requirements? How do these local functions converge? Do they all converge uniformly, at the same rate as the global function, or do they display non-uniform convergence behavior? Do a majority of them converge quickly, well before the convergence of the global objective function? These are some of the questions our empirical work will try to uncover.

At this time, we will empirically analyze, in some detail, our chosen example analytic—document clustering with k-means. We choose document clustering with k-means because it is arguably the first analytic that a user might want to run in a large number of enterprise applications. Most EIM vendors today offer the ability to cluster a user's data, but several applications which could potentially use this clustering do not do so since it takes considerably longer than real-time to finish clustering a large dataset. In summary, we are aware of several applications that need clustering, but currently rely on a static, older clustering of the data instead of allowing the user to dynamically cluster data as they proceed through their workflow. We show how our approach can mitigate this situation, and how we can instrument k-means with local user-centric functions to extract near real-time information that is useful to the user at their current stage of workflow.

3 Example: Document Clustering with k-means

The k-means algorithm is ubiquitous in data mining [10]. k-means can be used at various stages in EIM: to understand high-level organization of data [2, 5, 6], to organize search results [3], to extract semantic information such as labels [4], and so on. k-means is also time-intensive, and therefore a good candidate for us to demonstrate our approach.

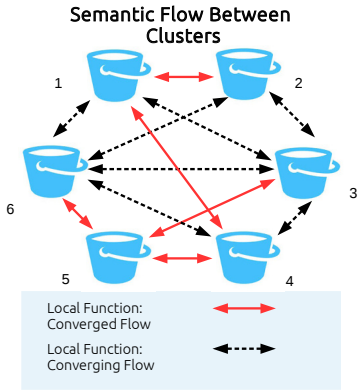

Figure 1: Schematic of the view of k-means through local functions: k-means is a set of pairwise flows, most of which abate early. Once flows to and from a cluster have abated—as has happened to Cluster 5—, we may extract semantic meaning from it. This can happen very early during run-time, long before final convergence of k-means.

3.1 Preliminaries

We briefly provide the framework of the document representation we use. Since we are clustering text, we use a tf-idf weighted bag-of-words vector representation for the documents [7]. We use a standard stoplist, and remove all words that occur fewer than three times in the corpus. As is standard, we normalize each vector to unit length so that if two documents of different lengths are still speaking of the same topics, they are regarded equally [8]. Finally, we use the cosine of the angle between the vectors as our similarity metric since it is known to outperform metrics such as Euclidean distance for text applications [9].

We use the random assignment version of k-means (as opposed to Forgy), where each document is randomly assigned a cluster at initialization. Cluster centroids are then computed, and re-assignment of documents to the closest (in terms of cosine similarity) cluster is done iteratively until there is no further movement of documents between clusters.

3.2 User-Centric Objectives: Concept Classes

A key component of our approach is to replace the focus on the global objective function with local user-centric functions. These functions should capture the domain-specific requirements of the user. What would such requirements be in the case of enterprise applications of document clustering?

A large majority of EIM applications that (could) use clustering want to understand the semantics, or "meaning" of each cluster. In other words, clustering is seen as

a technique that groups the data into conceptually coherent groups, each group speaking of a coherent class of concepts. These concepts are then used in the next stage of the EIM pipeline: for example, they may be used in scatter-gather type workflows (for example, some eDiscovery workflows), to create taxonomies (for records management or classification), and so on. Therefore, the first task in capturing the user's requirements is to compute, from each cluster, a set of coherent concepts that it speaks about.

3.2.1 Cluster Digests: Concept Labels

Definition 1 *Define D_i as the Boolean indicator variable for document inclusion into C_i. Define $t_{j,D}$ as the Boolean indicator variable for term inclusion in a document.*

$$D_i[D,C_i] = \begin{cases} 1, & \text{if } D \in C_i, \\ 0, & \text{else;} \end{cases} \quad t_{j,D}[t_j,D] = \begin{cases} 1, & \text{if } t_j \in D, \\ 0, & \text{else.} \end{cases} \tag{3}$$

Therefore, D_i is a random variable whose arguments are $[D,C_i]$. $t_{j,D}$ is a random variable with arguments $[t_j,D]$. Therefore, for a fixed i and j, both of these are random variables over the set of documents. In this case, their mutual information is well defined.

Definition 2 *We define $I[i,j]$ as the mutual information between the random variables D_i and $t_{j,D}$.*

Conceptually, this measures the increase in (conditional) probability of a document being placed by the k-means algorithm in cluster C_i given that it has the term t_j. In practice, we also perform thresholding: namely, we only count those terms that occur at least five times in the corpus in order to preclude terms that may occur only in a few documents, all of whom land in one cluster.

Definition 3 *For $\ell > 0$, the ℓ* concept labels *associated to cluster C_i are the top ℓ terms $\{t_j\}$ in the corpus in descending order of $I[i,j]$. We denote the set of concept labels for C_i by T_i.*

3.3 Local Functions: Concept Flows

In order to demonstrate our approach, we construct certain functions that can be measured retroactively: namely, measuring them requires the algorithm to have converged. However, the empirical properties of these functions will suggest that, indeed, these functions can be approximated in real-time.

At each iteration of the k-means algorithm, documents move between clusters. We wish to measure how much information that is core to the cluster enters and leaves each cluster as a result of this.

In order to measure this "concept flow" when a document moves between cluster C_{i_1} and C_{i_2}, we measure the presence of terms in the document that are concept labels for C_{i_1} and C_{i_2}. By taking the difference of these two quantities, we obtain a measure of the "concept flow" associated to the movement of the document.

We wish to measure the flow of concepts in both directions between a pair of clusters, at any iteration.

Definition 4 *Let $\ell > 0$ and $m < n$. Let document D move from C_{i_1} and C_{i_2} at iteration m of k-means. Let $n_{j,D}$ be the number of times term t_j occurs in document D. The* forward concept flow *associated with the document D at iteration m is defined as*

$$\sigma_f[D,m] := \sum_{t_j \in T_{i_1}} n_{j,D}. \tag{4}$$

The reverse concept flow *associated with the document D at iteration m is defined as*

$$\sigma_r[D,m] := \sum_{t_j \in T_{i_2}} n_{j,D}. \tag{5}$$

The total concept flow *associated with the document D at iteration m is defined as*

$$\sigma_t[D,m] := \sigma_f[D,m] - \sigma_r[D,m]. \tag{6}$$

We also define the net quantities obtained by summing the above over all documents that move from one cluster to another.

Definition 5 *The* net forward concept flow *from cluster C_{i_1} to cluster C_{i_2} at iteration m is defined as*

$$\Sigma_f[i_1,i_2,m] := \sum_{D \in C_{i_1}} \sigma_f[D,m]. \tag{7}$$

The net reverse concept flow *from cluster C_{i_1} to cluster C_{i_2} at iteration m is defined as*

$$\Sigma_r[i_1,i_2,m] := \sum_{D \in C_{i_1}} \sigma_r[D,m]. \tag{8}$$

The net concept flow *from cluster C_{i_1} to cluster C_{i_2} at iteration m is defined as*

$$\Sigma_t[i_1,i_2,m] := \sum_{D \in C_{i_1}} \sigma_t[D,m]. \tag{9}$$

Finally, we define the average per-cluster-pair quantities.

Definition 6 *The* average forward concept flow *between and ordered cluster pair at iteration m is defined as*

$$\overline{\Sigma}_f[m] = \frac{1}{k(k-1)} \sum_{i_1,i_2;i_1 \neq i_2} \Sigma_f[i_1,i_2,m]. \tag{10}$$

The average reverse concept flow *between an ordered cluster pair at iteration m is defined as*

$$\overline{\Sigma}_r[m] = \frac{1}{k(k-1)} \sum_{i_1,i_2:i_1 \neq i_2} \Sigma_r[i_1,i_2,m]. \quad (11)$$

The average (net) concept flow *between an ordered cluster pair cluster at iteration m is defined as*

$$\overline{\Sigma}_t[m] = \frac{1}{k(k-1)} \sum_{i_1,i_2:i_1 \neq i_2} \Sigma_t[i_1,i_2,m]. \quad (12)$$

Notice that although we are measuring the semantic flow, as described above, during pre-convergence iterations of k-means, we obtain the labels only after convergence. Let c denote the iteration at which k-means converges. Then, measurement of semantic flows, with respect to the final labels, at iteration $m(< c)$ requires us to wait until convergence at iteration c. However, let us now inspect these flows carefully, and see if we may approximate them in real-time.

4 Experimental Results

4.1 Datasets and Protocol

We used two standard benchmark datasets for document clustering. The first is `N20`, the 20 Newsgroups dataset that contains roughly 20,000 articles posted to 20 usenet group. The articles are more or less evenly divided between the newsgroups; however some newsgroups are highly related, while others are not. The second dataset is `REU`, the Reuters-21758 dataset that has documents from Reuters newswire having 82 primary topics. For both datasets, we ran k-means with the "natural" number of clusters k—namely, 20 for `N20`, and 82 for `REU`.

We ran each clustering experiment five times. Since our results require us to examine concept flows between specific pairs of clusters, and these pairs change from experiment to experiment, we picked the experiment that was most typical of the five (in terms of convergence behavior) to depict our results. The variance between experiments was minor, and the form of the results did not change from experiment to experiment.

For the most typical experiment (as described above), the clustering of `N20` took 35 iterations, while that for `REU` took 52 iterations. For each experiment, we ordered the $k(k-1)$ ordered pairs of clusters by descending order of semantic flows, summed over iterations $[10, 15]$. Next, we measured the changes in semantic flow for all these pairs as the experiment progressed. Fig. 2 shows results for `REU`.

4.2 Properties of Concept Flows

The inspection of the graphs in Fig. 2, and the similar graphs for `N20` (which we could not show due to lack of space) immediately lead us to the following empirical result:

1. Local functions, unlike objective functions, are not monotone. The sequence $\Sigma_t[i_1,i_2,1], \Sigma_t[i_1,i_2,2], \ldots$ shows a zig-zag behavior until it falls to zero.
2. The average flow first rises sharply, but then starts to fall sharply after only a few (less than 5) iterations for both datasets. Compare this to the convergence time for each dataset (52 and 35 iterations, respectively).
3. The average reverse flow has also nearly abated by this time (i.e., by 5 iterations).
4. In the few cases of cluster pairs where flows are significant even after they have abated in other pairs; we found that the clusters themselves are semantically related.

These empirical results, repeated over multiple experiments, suggest that for a large majority of clusters, the "documents that matter" have already been placed into their correct clusters well before final convergence of k-means. Thus, our flow measurements uncover an "almost everywhere convergence" of k-means well before it converges globally in terms of its objective function.

We have experimented with other values for k, and the results are similar.

4.3 Near Real-Time Information

At this time, we can answer the question "what information can be extracted in near real-time as a result of the properties in §4.2?"

We have seen, empirically, that local flow functions for a majority of cluster pairs abate very quickly—between 5 and 10 iterations. At this time, we can extract concept classes for each cluster. For a majority of the clusters, these concept classes will continue to be accurate at convergence. The few clusters where these concept classes change significantly can be detected by our local concept flow function measurements, and updated accordingly. In this manner, we can already provide the user with a large proportion of the information that they desired from the analytic, but well before the analytic actually converges. In cases where k-means takes of the order of a few minutes to complete, the time taken to provide this information will be of the order of (tens of) seconds, which can enable a near real-time workflow.

The key idea behind our approach is to use the "almost everywhere convergence" to start providing local information to the user at places where such convergence has already happened, and not wait for global convergence.

In general, in any workflow where each cluster has to be further examined, we can supply the user with information on all the clusters that have already converged, so that they can begin examining those. This yields several examples of enterprise workflows where local information gathered as described above can enable real-time workflows. One example is a large eDiscovery workflow.

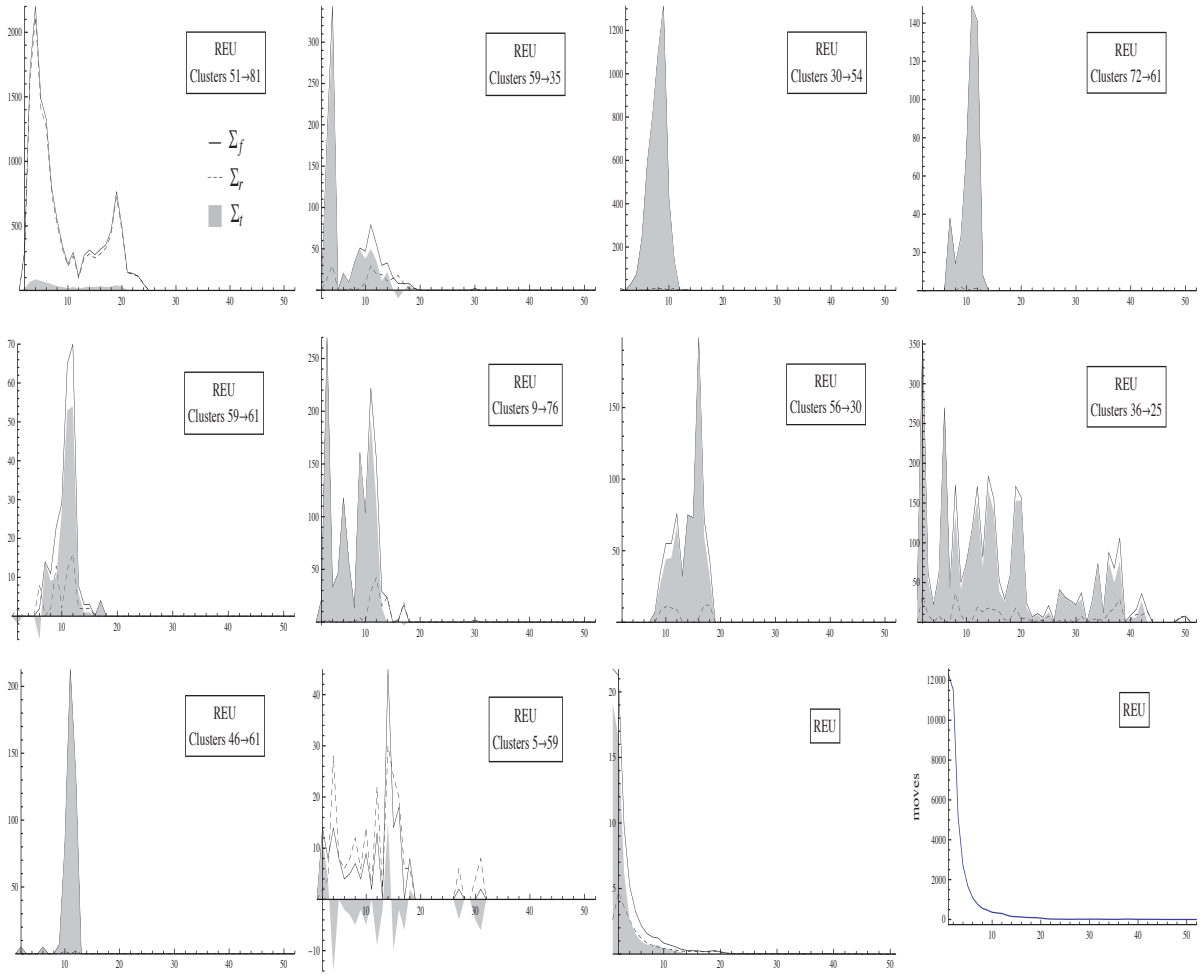

Figure 2: The first ten graphs show the top ten semantic flows for a run of k-means on REU. The Y-axis measures flows. A majority of the lower-ranked flows abate much quicker, and even the highest ranked flows tend to abate well before convergence. Legends are shown only on the first graph. The next graph is the average flow, taken over all pairs (not just the topmost). The final graph is the total number of documents that move at each iteration of k-means. In all graphs, the X-axis is the iteration number. For lack of space, only the experiments on REU are shown; similar results were observed on N20.

Mid-sized eDiscovery cases frequently have to examine a few hundred thousands of documents in a limited time-frame. For example, in early case assessment, this time frame might be only a few weeks. If clustering is used to organize the inspection of these documents, then the inspection of clusters that have already converged can begin as soon as their information is available, without waiting for clustering to converge throughout the corpus. The larger the corpus of documents, the more time is saved using this real-time enabled workflow, over a workflow where the user waits for corpus-wide convergence.

In any scatter-gather workflow [2], the user examines each cluster individually during the gather phase, and decides whether it should be included in the subsequent scatter phase. This represents another generic workflow where providing clusters as soon as they have converged can enable the use to make their decision on the available clusters, without waiting for the remaining clusters.

A natural question that may be asked is: can a cluster where convergence seems to have happened "change" its convergent state? Can it start showing an increased flow after having seemingly converged? First of all, we must ensure that the flow has indeed abated for a period of a few successive (say, 5) iterations. We did not observe significant flows after a abatement of flows lasting five iterations. Rarely, we do see a small additional flow in such cases, for example, the flow $5 \rightarrow 59$ shown in Fig. 2, but as is the case in the example, it is not very large.

What about pairs of clusters where flow has not abated till a relatively late stage? We can simply flag such pairs of clusters as being "semantically related," to be consid-

ered together for scatter-gather. This accurately reflects, for instance, eDiscovery workflows.

4.4 Time to Measure Local Functions

With the following pragmatic choices, we can instrument k-means for real-time local flow functions at insignificant additional cost.

1. In order to measure the local functions $\Sigma_t[i_1, i_2, 1]$, we need $k(k-1)$ counters, one for each bidirectional measurement of flow between each pair of clusters. The time taken to updating each counter is dominated by the time to compute the actual move to be made for each document, and does not significantly increase their sum.

2. Moreover, this need not be done at every iteration, since all we want is to detect abatement of flows. We experimented with computing flow only at iteration 5, 10, 20, 30, and so on, yielding satisfactory results.

3. The time taken to compute labels can be significant; however, in light of the quick abatement of flows, we can compute these labels only once, soon after iteration 5.

5 Related Work

We are not aware of any work that studies the behavior of k-means with respect to local user-centric functions. However, more generally, our work may be seen as a study of the k-means algorithm during its convergence. In this regard, the work that is closest to ours is [1]. However, there are obvious and fundamental differences: besides the core difference of local vs. global functions, [1] studied the convergence of k-means on the IRIS dataset, which has only four dimensions. One of the primary properties of text document corpora that distinguish it is the high-dimensional and sparse nature of the feature vectors. As expected given these important differences, the results of the experiments (namely, the trajectories of the functions under study) vary greatly. As but one example, the behavior of the sequence $\Sigma_t[i_1, i_2, 1], \Sigma_t[i_1, i_2, 2], \ldots$ is very dissimilar to that of objective function values.

6 Conclusion and Future Work

We have demonstrated that time-intensive analytics such as clustering can be calibrated to yield information in near real-time due to an empirically observed almost-everywhere local convergence property. This real-time information can enable users to conduct their workflows without waiting for the analytic to converge everywhere.

This work was motivated by real-world applications of clustering in EIM. In particular, we are intrigued by the possible applications of the techniques of this paper to cluster-based retrieval over large document corpora. Abstractly, we have a ranking of clusters based on their convergence. We also have a retrieval ranking of clusters based on their relevance to some information need. Can a meaningful merger of these two rankings be done to provide the user with the most relevant information to their need, as quickly as it is available?

Acknowledgement

Some of the data in this paper was gathered using scripts written by Hernan Laffitte.

References

[1] BOTTOU, L., AND BENGIO, Y. Convergence properties of the k-means algorithm. In *Advances in Neural Information Processing Systems: Proceedings of the 1994 Conference* (Cambridge, Massachusetts, 1995), G. Tesauro, D. Touretzky, and T. Leen, Eds., MIT Press, pp. 585–592.

[2] CUTTING, D. R., PEDERSEN, J. O., KARGER, D., AND TUKEY, J. W. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1992), pp. 318–329.

[3] HEARST, M. A., AND PEDERSEN, J. O. Re-examining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval* (New York, 1996), ACM Press, pp. 76–84.

[4] KARYPIS, G., AND HAN, E.-H. S. Fast supervised dimensionality reduction algorithm with applications to document categorization & retrieval. In *Proceedings of the ninth international conference on Information and knowledge management* (New York, NY, USA, 2000), CIKM '00, ACM, pp. 12–19.

[5] LAGUS, K., HONKELA, T., KASKI, S., AND KOHONEN, T. Self-organizing maps of document collections: A new approach to interactive exploration. In *KDD* (1996), pp. 238–243.

[6] PIROLLI, P., SCHANK, P., HEARST, M., AND DIEHL, C. Scatter/gather browsing communicates the topic structure of a very large text collection. In *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems* (1996), vol. 1 of *PAPERS: Interactive Information Retrieval*, pp. 213–220.

[7] SALTON, G., AND MCGILL, M. Introduction to modern information retrieval. McGraw-Hill, 1983.

[8] SINGHAL, A., BUCKLEY, C., MITRA, M., AND SALTON, G. Pivoted document length normalization. Technical Report TR95–1560, Department of Computer Science, Cornell University, Nov. 1995.

[9] STREHL, A., GHOSH, J., AND MOONEY, R. J. Impact of similarity measures on web-page clustering. In *Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin* (July 2000), AAAI/MIT Press, pp. 58–64.

[10] WU, X., KUMAR, V., ROSS QUINLAN, J., GHOSH, J., YANG, Q., MOTODA, H., MCLACHLAN, G. J., NG, A., LIU, B., YU, P. S., ZHOU, Z.-H., STEINBACH, M., HAND, D. J., AND STEINBERG, D. Top 10 algorithms in data mining. *Knowl. Inf. Syst. 14*, 1 (Dec. 2007), 1–37.