Multi-Level Clustering Technique Leveraging Expert Insight

Manu Kuchhal[*] Ritwik Chaudhuri[†] Sudhanshu Singh[‡] Sarthak Ahuja [§]
Gyana Parija[¶]

Abstract
State of the art clustering algorithms operate well on numeric data but for textual data rely on conversion to numeric representation. This conversion is done by adopting approaches like TFIDF, Word2Vec, etc. and require large amount of contextual data to do the learning. Such contextual data may not be always available for the given domain.

We propose a novel algorithm that incorporates Subject Matter Experts' (SME) inputs in lieu of contextual data to be able to do effective clustering of a mix of textual and numeric data. We leverage simple semantic rules provided by SMEs to do a multi-level iterative clustering that is executed on the Apache Spark Platform for accelerated outcome. The semantic rules are used to generate large number of small sized clusters which are qualitatively merged using the principles of Graph Colouring. We present the results from a **Recruitment Process Benchmarking** case study on data from multiple jobs. We applied the proposed technique to create suitable job categories for establishing benchmarks. This approach provides far more meaningful insights than traditional approach where benchmarks are calculated for all jobs put together.

Key Words: clustering, recruitment process benchmarking, spark

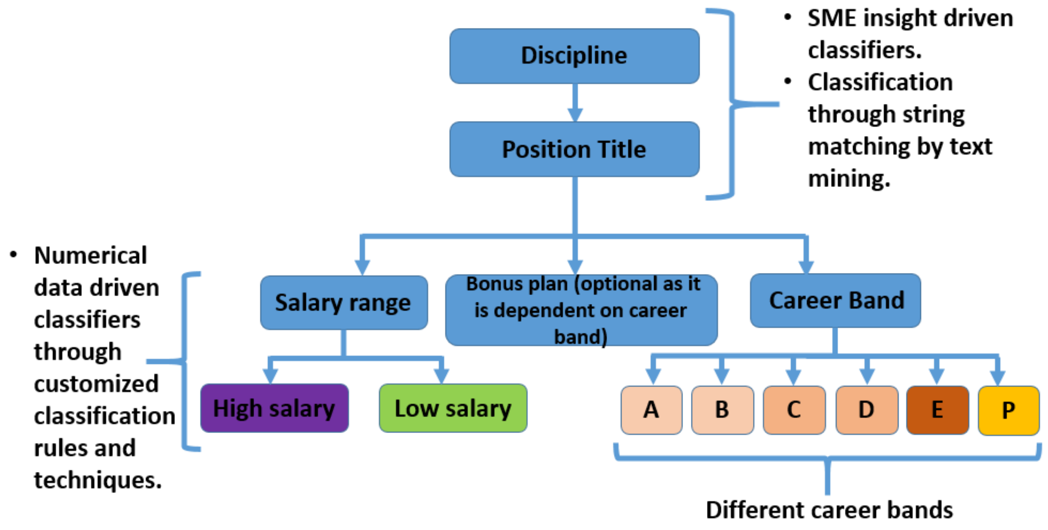
1. Introduction

In industry and academia, recruitment of new candidates is often a prolonged process, which usually involves multiple steps - creating a job requisition, approving the requisition, collecting applications, manual screening of applications to select a smaller subset of applicants, interviewing applicants, and finally making a decision to form a list of candidates to whom the offers are extended. Since the recruitment process is divided into several stages, any delay incurred in one of these stages culminates into a delay in the overall recruitment process. To prevent such delays, there are industry benchmarks set for each company, but more often than not are failed to be met. An interesting feature about these benchmarks is that are they are *uniform over all job families* in the industry leading to cases where for some job positions, these benchmark thresholds are met comfortably while for some others, they turn out to be too aggressive. For example, the duration of the recruitment process for an entry level position, will be naturally much lesser than that for a senior level position. Hence, one benchmark threshold set over all jobs does not suffice, making it important to have different benchmarks for similar jobs within a particular industry. In order to set the described benchmarks, it is important to determine which jobs positions fall within the same group. The first part of our work is precisely based on classifying these similar jobs into clusters.

Clustering jobs into similarity cluster is a non-trivial problem. In an industry, there could be jobs within a discipline which fall in different classes but also jobs from different disciplines which are similar to each other and hence should be analyzed together. A job

[*]IBM India Pvt. ltd.
[†]IBM India Pvt. ltd.
[‡]IBM India Pvt. ltd.
[§]IBM India Pvt. ltd.
[¶]IBM India Pvt. ltd.

is defined by several parameters such as job description, position title, career band level, bonus type, discipline, business group, salary range, geo etc. A number of classical clustering techniques such as K-means, K-medoids, density based clustering techniques such as DBSCAN, can be tried out to cluster jobs, and might even produce a reasonable value of cluster quality metric, but in our practical experience SMEs tend to disagree with the results because they do not concur with business intelligence/SME insight. For rest of the paper, our discussion is based on data and SME insight collected from an industry standard applicant tracking system.



Discipline
Position Title
• SME insight driven classifiers.
• Classification through string matching by text mining.
• Numerical data driven classifiers through customized classification rules and techniques.
Salary range
Bonus plan (optional as it is dependent on career band)
Career Band
High salary
Low salary
A
B
C
D
E
P
Different career bands

Figure 1: SME suggested hierarchy of classifiers.

As shown in Figure 1, SME suggested hierarchy for job clustering include discipline, position title, salary range, career band and bonus type in the descending order of importance. The use of bonus plan as an attribute remains optional as it is dependent on Career Band.

The novelty of our work is based on two fundamental and efficient customized clustering techniques which can be used to cluster similar kind of jobs into similar categories. We test our methodology on a real data set and carry out a qualitative analysis of the results obtained by seeking feedback from the SME. We find the results from our clustering approach to be more in sync with the SMEs understanding than any of the classical clustering techniques mentioned earlier.

The complexity of the customized algorithm grows with the number of job families and also with the number of different jobs discipline and job positions. Essentially, if the number of different jobs are n, then the computation complexity of the customized algorithm is $O(n^4)$. Hence for large enough n, which is usually the case, it is essential to parallelize the customized algorithm in an efficient manner to execute the clustering methodology in a reasonable time frame. The parallelization of the algorithm is done leveraging Spark capabilities. Spark essentially parallelizes the basic operations such as addition, subtraction, multiplication, division, counting, returning query results in an efficient inherent way when the data is stored in the form of a Resilient Distributed Datasets (RDDs). Once the dataset is distributed in RDDs and computed results are obtained from every parallelized nodes, the results are collated together to obtain the final desired result. Due to heavy computation, the entire data was broken into 16 different parallel nodes. The algorithm can be parallelized into a larger number of nodes but one needs to be careful about the time taken to collect and collate the results together at the end for each node.

To evaluate how good the clustering algorithm has performed, we have also devised an evaluation methodology which evaluates the similarity of jobs on different attributes of the data. For example, ideally one would expect jobs with same position title, same discipline, same salary range, same bonus plans, same bands, etc. to be clustered together. But rarely, such an ideal cluster is formed. Hence, in a cluster there could be different position titles arising from different disciplines with different salary range, bonus plans, salary bands. To evaluate the overall similarity of the jobs in the cluster, it is important to measure the similarity of jobs under some metric over all the stated attributes, generating multiple cluster scores. Based on these cluster scores of jobs in each cluster, one can come up with an overall measure such as weighted mean of all the cluster score. The usual single number measure for cluster performance does not produce reliable results as it does describe the information obtained from various attributes associated with the jobs which got clustered together.

In this paper we suggest data driven customized methodologies for

- (A) Clustering similar jobs into similar categories based on SME insights
- (B) Merging of clusters (formed in (A)) based on different attributes
- (C) Leveraging Spark capabilities to parallelize each clustering algorithm
- (D) Evaluating the performance of the clustering methodology

The novelty of the proposed methods can be used in different domains of clustering problems where some sort of prior information on the similarity of the data instances is provided to supervise the clustering technique. Also this particular methodology can be used in recommender systems where different kind of objects such as clothes, electronic goods, children toys are categorized into different categories. Clothes, electronic goods, children toys can be thought of as a part of different type of commodity families but based on customer liking, within each of these families further clustering is needed so that similar kind of commodities are recommended to the customer.

Towards the end of this paper, we move towards the second part of our work- a detailed case study, where we come up with different benchmarks for number of days taken to close similar job requisitions corresponding to each cluster. Also, to further analyze the delay in different stages of recruitment process we come up with benchmarks for the following and share the inferences as a part of this report -

- Number of days to close a requisition
- Number of days between approval of requisition to opening to the requisition
- Number of days between creation of requisition to approval of the requisition
- Proportion of applications received for a requisition to screened
- Proportion of candidates called for interview to screened
- Proportion of candidates rejected in the interview to number of candidates called for the interview
- Proportion of candidates rejected an offer to the number of candidates to which offers were extended

2. Literature Review

In industry, benchmarking has been implemented way before the concept of benchmarking was popularized and theroized in academic research. Kleinhas, Merle and Doumeingts (1995) discussed, the benchmarking to be the ongoing task at all levels in the business to improve on the performance that deliver customer satisfaction. In the context of recruitment the benchmarks are analyzed to improve on the entire hiring process so that poorly affected sub processes are identified and the recommendation for improvement is proposed. In the same paper, Kleinhans, Merle and Doumeingts (1995) suggested that internal analysis of an organization helps to identify the processes and functions that need improvement. This concept is applicable as it is in the context of recruitment process. Performance based clustering methods have been tried in different domains such as benchmarking the performance of airport operations. Sarkis and Talluri (2004) described how to benchmark the performance of airports in their work. Also performance evaluation and improvement of airport operations largely improve the customer base to attract more passengers and in turn it improves the benefits of the stakeholders. In their work, Sarkis and Talluri (2014) used data envelopment analysis (DEA) methods for determining the efficiencies of airport operations and then clustered the results to identify the benchmarks for poorly performing airports and what are the opportunities of improvement. Dai and Kuosmanen (2014) use clustering techniques to identify decision making units (DMU) which are similar in terms of input - output profiles or other observed characteristics. They rank DMUs within each cluster, and rank clusters based on their overall efficiency. This helps in identifying the most efficient benchmarks, and the top performers within clusters as well. They apply their methodology to the regulation of electricity distribution networks in Finland, where the regulator uses the semi-nonparametric StoNED method (stochastic non-parametric envelopment of data). Sharma and Yu (2009) use physical values that represent relevant properties of the terminals to club similar terminals together to establish efficiency benchmarks. They fuse data mining and DEA to provide a diagnostic tool to effectively measure the efficiency of inefficient terminals and prescribe actions to reach the efficient frontier in accordance with their maximum capacity and similar input properties which otherwise is not possible with DEA alone.

Since our focus in this paper is on use of custom clustering technique, reporting and comparing results of various clustering schemes to identify the best one is also an aspect we discuss in this paper. In this regard, both external and internal measures of strength of clustering are available in literature. Given the context of the broader problem that we address here, only internal measures are relevant here. DaviesBouldin index (Davies and Bouldin, 1979), Dunn index (Dunn, 1973) and silhouette index (Rousseeuw, 1987) are some of the most popular internal clustering measures used. In some relatively recent work, Meila (2005) describe a criterion for comparing clusterings by viewing them as elements of a lattice, the natural algebraic structure for the partitions of a set. Achtert et al (2012) describe a visual representation of clusterings based on pair counting.

3. Basic Methodology

We first describe the methodology for clustering jobs based on SME inputs.

3.1 Clustering Similar jobs based on SME inputs

Jobs are first clustered based on discipline and position title pertaining to a job. E.g., the disciplines ’Administrative’ and ’Administrative Finance’ are clustered together based on the fact that the disciplines are similar. On the other hand a discipline such as 'Business Management' is assigned to a different cluster from the above one because the discipline is different from the ones stated above. The variable 'discipline' is chosen as the first variable of clustering based on the SME insight. In a general context where a hierarchy of the variables are defined before hand based on the SME input, the top level variables are considered to be the initial classifiers. In this particular context, the SME insight was to first cluster the jobs which have similar job disciplines. Below, we describe a *string matching algorithm* using which a variable 'discipline' , referred to as v_1 henceforth, can be used for clustering. First the entries in v_1 are sorted in the ascending order of alphabets. Then a bag of key words $\mathcal{K}_{v_1}$ are obtained pertaining to the variable v_1. The keywords are based on SME inputs. If SME inputs for the keywords are not provided, then one can use text mining techniques to form a bag of key words. For example in this context 'Administrative' is a key word and belongs to $\mathcal{K}_{v_1}$. In the next step, to check if the i^{th} entry of v_1 denoted by v_1^i and $(i+1)^{th}$ entry of v_1 denoted by v_1^{i+1} belong to the same cluster or not, the words forming v_1^i are denoted as $\mathbf{w}^i$ and analogously for v_1^{i+1} as $\mathbf{w}^{i+1}$. So, for example if v_1^i is 'Administrative Finance' then $\mathbf{w}^i = \{$'Administrative', 'Finance'$\}$. Now the similarity of v_1^i and v_1^{i+1} are calculated in the following manner

$$S_{i,i+1} = \frac{|\mathbf{w}^i \cap \mathbf{w}^{i+1} \in \mathcal{K}_{v_1}|}{\max(|\mathbf{w}^i \in \mathcal{K}_{v_1}|, |\mathbf{w}^{i+1} \in \mathcal{K}_{v_1}|)}$$

Where $|A|$ denotes cardinality of A. If

$$S_{i,i+1} >= C_{\max(|\mathbf{w}^i \in \mathcal{K}_{v_1}|, |\mathbf{w}^{i+1} \in \mathcal{K}_{v_1}|)}, \tag{1}$$

where $C_{\max(|\mathbf{w}^i|, |\mathbf{w}^{i+1}|)}$ is a pre-defined threshold depending on the maximum number of words in either of the strings, then v_1^i and v_1^{i+1} are classified in the same cluster. If the number of data points in v_1 is n, then in the above algorithm i is varied in range $1, 2, 3, \ldots, n-1$. After first level clustering is done, if needed, further clustering can done based on the other variables. For this particular problem, 'position title' was taken as the second level clustering variable, referred to as v_2 henceforth. SME insight was given as 'discipline' to be taken as the primary classifying variable and 'position title' to be taken as the secondary classifying variable. Hence, following an analogous approach the entries within each first level cluster pertaining to the variable v_2 are sorted in the ascending order of alphabets. Then, a bag of key words $\mathcal{K}_{v_2}$ are obtained pertaining to the variable v_2. For the entries in each first level cluster, the same string matching algorithm as stated above was applied for the variable v_2. Thus the entries belonging to same first level clusters are further clustered to form a set of second level clusters. In a general context, if more variables $v_3, v_4 \ldots$ are stated as the classifiers then more nested clusters can be formed and named as third level cluster, fourth level cluster etc. For this problem, there were no more classifiers given by the SME.

Clustering the data based on SME input may give rise to a huge number of clusters due to the presence of many different disciplines and position titles in the data. Also, it is possible that due to the similarity among other variables such salary range, salary band, data points can be clustered as similar. In general, in the context of recruitment process benchmarking, usually it is found that jobs which belong to the same salary range and same band level are very similar to each other. Hence, to bring this aspect into the clustering algorithm, clusters formed in (3.1) are merged based on the attributes such as salary range and salary band. Merging of clusters also reduces the number of overall clusters. For this particular problem, the merging of cluster was done based on the variable salary range and salary band. But the proposed algorithm for merging of clusters can be used on any other variables in the context of some other problem.

3.2 Merging of clusters (formed in (A)) based on different attributes

SME based clustering techniques may give rise to a very large number of job clusters if the variables v_1, v_2, ..., v_n have many unique values. As mentioned earlier, in the context of recruitment process benchmarks, jobs with similar salary range and bands can be considered similar and should be clustered together. The process of merging the clusters based on salary range and band is essentially re-clustering of already created clusters. It reduces the number of clusters formed in (A) and also increases similarity among jobs within a cluster. For merging of clusters, two different approaches are tried out:

- Sequential merging approach
- Clique-based approach

These merging approaches require that a proximity measure be defined for clusters.

Methodology for computing distance between clusters A and B: Let us assume there are n_A elements in cluster A and n_B elements in cluster B. For this particular problem the distance between A and B are computed based on the similarity of high salary average, low salary average and proportion of career bands in A and B. Let us assume high salary amount and low salary amounts for each of the elements in cluster A are denoted by S_H^i and S_L^i for all $i \in \{1, 2, \ldots, n_A\}$. Analogously for cluster B, the high salary amount and low salary amounts are denoted by S_H^i and S_L^i where $i \in \{1, 2, \ldots, n_B\}$. Also, let us assume there are K different salary bands and the proportional distribution of salary bands in cluster A is P_A^i for all $i \in \{1, 2, \ldots, K\}$. So, $\sum_{i=1}^{K} P_A^i = 1$. Analogously, the proportional distribution of salary bands in cluster B is P_B^i for all $i \in \{1, 2, \ldots, K\}$. Then, the distance D_{AB} between the cluster A and B is computed as:

$$\begin{aligned} D_{AB} &= \frac{1}{4}\Bigg(\frac{|\frac{1}{n_A}\sum_{i=1}^{n_A} S_H^i - \frac{1}{n_B}\sum_{i=1}^{n_B} S_H^i|}{\max(\frac{1}{n_A}\sum_{i=1}^{n_A} S_H^i, \frac{1}{n_B}\sum_{i=1}^{n_B} S_H^i)} \\ &+ \frac{|\frac{1}{n_A}\sum_{i=1}^{n_A} S_L^i - \frac{1}{n_B}\sum_{i=1}^{n_B} S_L^i|}{\max(\frac{1}{n_A}\sum_{i=1}^{n_A} S_L^i, \frac{1}{n_B}\sum_{i=1}^{n_B} S_L^i)} \\ &+ \sum_{i=1}^{K} |P_A^i - P_B^i|\Bigg) \end{aligned} \tag{2}$$

Where $|x|$ denotes the absolute value of the number x. Similarity S_{AB} between A and B is computed as

$$S_{AB} = 1 - D_{AB}. \tag{3}$$

The scaling parameter $\frac{1}{4}$ is used in equation (2) because the maximum value of expression within the parenthesis in equation (2) is 4 (as can be seen below),

$$\frac{|\frac{1}{n_A}\sum_{i=1}^{n_A} S_H^i - \frac{1}{n_B}\sum_{i=1}^{n_B} S_H^i|}{\max(\frac{1}{n_A}\sum_{i=1}^{n_A} S_H^i, \frac{1}{n_B}\sum_{i=1}^{n_B} S_H^i)} \leq 1$$

$$\frac{|\frac{1}{n_A}\sum_{i=1}^{n_A} S_L^i - \frac{1}{n_B}\sum_{i=1}^{n_B} S_L^i|}{\max(\frac{1}{n_A}\sum_{i=1}^{n_A} S_L^i, \frac{1}{n_B}\sum_{i=1}^{n_B} S_L^i)} \leq 1$$

$$\sum_{i=1}^{K} |P_A^i - P_B^i| \leq \sum_{i=1}^{K} |P_A^i| + |P_B^i| = \sum_{i=1}^{K} (P_A^i + P_B^i) = 2$$

Note that the distance/similarity values computed in this fashion are always between 0 and 1.

Sequential merging approach: In sequential merging approach, clusters are merged based on similarity scores. This score based merging is done sequentially over the clusters till there are no clusters left which are similar to each other. Essentially in sequential merging approach, the similarity of each pair of clusters are compared. As it can be seen in Figure 2, among four clusters A,B,C and D, the similarity between A and B is 0.9 and it is the maximum among all the other similarity scores between pairs of clusters. Hence A and B are merged together to form a bigger cluster denoted by (A,B). Then, in the next step the similarity of (A,B) and C, (A,B) and D, and C and D is calculated and it is found that the cluster (A,B) has maximum similarity with C and hence these two clusters are merged to form a bigger cluster denoted by (A,B,C). Finally the similarity of the cluster (A,B,C) and D is calculated and the similarity is found to be 0.1 which is less than the predefined threshold for merging and hence (A,B,C) and D are not merged together.

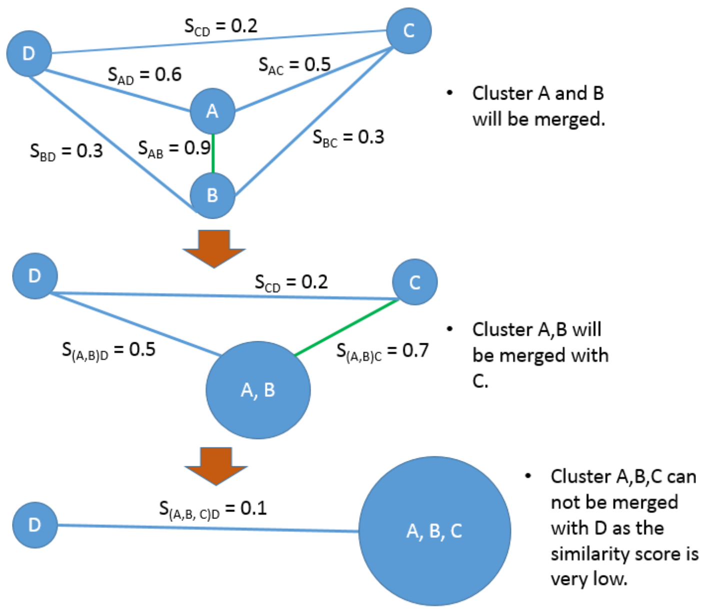


Figure 2: Sequential merging of four clusters A,B,C and D.

Clique-based approach: Let V be the set of all the labelled clusters (in first level or in second level). Now, we define a graph $G = (V, E)$ such that there is an edge between vertex A and vertex B if $D(AB) \leq T_0$ where $D(AB)$ is the distance between the vertex A and B and T_0 is a predefined threshold. Also note that $D_{AB} = 1 - S_{AB}$ where S_{AB} is the similarity score of clusters A and B. In this approach, instead of merging two closest vertices, we are trying to merge a whole class of vertices together which are very close

(i.e. cluster which are very similar), in some natural sense. We propose to merge all the independent maximal cliques in the graph formulated using clusters as the nodes and the distance between them as the edges. All the nodes with a distance within a defined threshold are considered to have an edge across them. Moreover, as proposed in Jain, Murthy and Flynn (1999), Jain and Dubes (1988) we plan to merge only one maximal clique per iteration in order to inculcate greediness in the merging algorithm. The following hierarchical preferences are adopted for merging the cliques:
1. The cliques with the maximum number of nodes are given preference.
2. In case of multiple cliques with the same number of nodes, the clique with the lowest average distance computed over all edges is preferred.
3. In case of multiple cliques with the same number of nodes and the same average distance, the clique with the lowest diameter will be preferred. Diameter is assumed to be the maximum distances between any two nodes in the clique in the current case.
4. If the ambiguity could still not be resolved, a random merge is undertaken.
In Figure 3, the clique based approach is explained with six nodes A,B,C,D,E and F. First, the edge length is calculated between every pair of nodes and the edge is kept only if the edge length is less than a certain threshold T_0. In the Figure 3, it can be seen that there are no edges between A and F, B and F, E and D, F and D since the distance between the nodes are larger than the pre-specified threshold. Once the edges are created in the graph, we search for the maximum sized clique. It can be seen that the maximum sized clique is formed by the nodes A,E,C and B. Hence A,E,C and B are merged together to form a bigger cluster denoted by (A,E,C,B). Now there is only one edge that is less than the threshold T_0 after the merging of A,E,C and B and that is the edge between node D and (A,E,C,B) and hence these will also be merged to form the final bigger cluster (A,E,C,B,D).

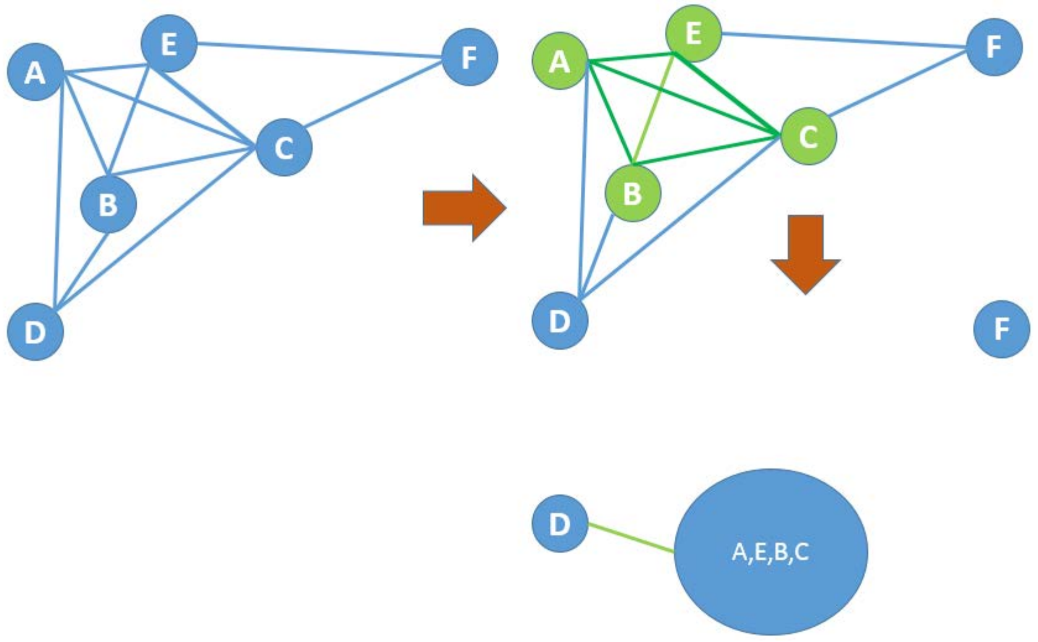


Figure 3: Clique based merging of six clusters A,B,C,D, E and F.

Clique based approach is a generalization of sequential merging approach. One of the important advantage of clique-based approach is all the clusters which are closely-knit (i.e. highly similar) to each other are merged together. In that sense, it is superior to sequential merging approach because in sequential merging approach it may happen that after merging two similar clusters, the overall similarity score of the newly formed cluster with some other cluster may decrease which might not have happened if all three clusters were merged together. Hence, sequential merging technique may ultimately produce suboptimal clustering quality. On the other hand the disadvantage of clique-based approach is that it is computationally very expensive as it involves the step of finding the maximum sized clique repeatedly. But the heavy computation yields better results. For huge datasets a hybrid approach may help in obtaining the results in a reasonable time frame.

For both clique-based and sequential merging approaches, clusters were merged in two different steps:

1. Intra level merging
2. Inter level merging

Intra level merging: In intra level merging, we look at all the second level clusters in each first level cluster and we merge them if they are sufficiently similar with respect to salary range (i.e. low salary, high salary) and bands. This requires calculating similarity scores between all pairs of second level clusters within each cluster. This is repeated unless all the distinct clusters are sufficiently dissimilar. That is, the similarity between each pair of clusters is less than the pre-specified similarity threshold. For example, assume that there are n first level clusters denote by $C_F^1, C_F^2, \ldots, C_F^n$. Now, fix $i_0 \in \{1, 2, \ldots, n\}$. For $C_F^{i_0}$, assume that there are n_{i_0} second level clusters denoted by $C_S^1, C_S^2, \ldots, C_S^{n_{i_0}}$. For each $i \neq j \in \{1, 2, \ldots, n_{i_0}\}$ the distance $D_{C_S^i C_S^j}$ between C_S^i and C_S^j is computed using equation (2). Similarity score $S_{C_S^i S_S^j}$, between C_S^i and C_S^j is calculated using equation (3). In total $\binom{n_{i_0}}{2}$ distances and similarity scores will be calculated.

- For sequential merging approach, among the $\binom{n_{i_0}}{2}$ similarity scores, the pair corresponding to maximum similarity score (if it is above a pre defined threshold) is merged. Ties are resolved randomly. With newly formed clusters in the first level cluster $C_F^{i_0}$, again the distances and similarity scores are re-calculated and the above process is repeated until there are no pair of second level clusters in $C_F^{i_0}$ having similarity score more than $S_{threshold}$.
- For clique-based approach, among $\binom{n_{i_0}}{2}$ distances, only those pairs of clusters are chosen which are at most T_0 apart. Here T_0 is the distance threshold between the two clusters. An edge is drawn between each pair of such clusters, with the clusters as nodes of the graph. Finally after forming the graph with edges, hierarchical order of merging cliques is followed as explained in the description of clique based approach. The clique size can be 2 also. With newly formed clusters in the first level cluster $C_F^{i_0}$, the new distances and similarity scores are recalculated and the above process is repeated until there are no cliques formed, out of second level clusters in $C_F^{i_0}$ with distance threshold less than T_0.

For both sequential merging approach and clique based approach the intra level merging is done for all first level clusters.

Inter level merging: In inter level merging we merge second level clusters, which are sufficiently close, across different first level clusters. Continuing with the same notation as defined in intra level merging, assume that for each of first level cluster $C_F^i \in \{1, 2, \ldots, n\}$ there are $C_S^{m_i}$ second level cluster. Now for each $i \neq j \in \{1, 2, \ldots, n\}$, the second level clusters in C_F^i and C_F^j are considered. Let these second level clusters be denoted by $C_{S_i}^1, C_{S_i}^2, \ldots, C_{S_i}^{m_i}$ and $C_{S_j}^1, C_{S_j}^2, \ldots, C_{S_j}^{m_j}$. For all $k \in \{1, 2, \ldots, m_i\}$ and $l \in \{1, 2, \ldots, m_j\}$ the distance between $C_{S_i}^k$ and $C_{S_j}^l$ is computed. Hence, for fixed first level clusters C_F^i and C_F^j there will be $m_i m_j$ distances between the second level clusters, one belonging to one first level cluster and the other belonging to the other first level cluster. The distances between all pairs of second level clusters belonging to all pairs of different first level clusters are calculated. In total, $\prod_{i=1}^{n} m_i$ distances and similarity scores between the pairs of second level clusters are calculated.

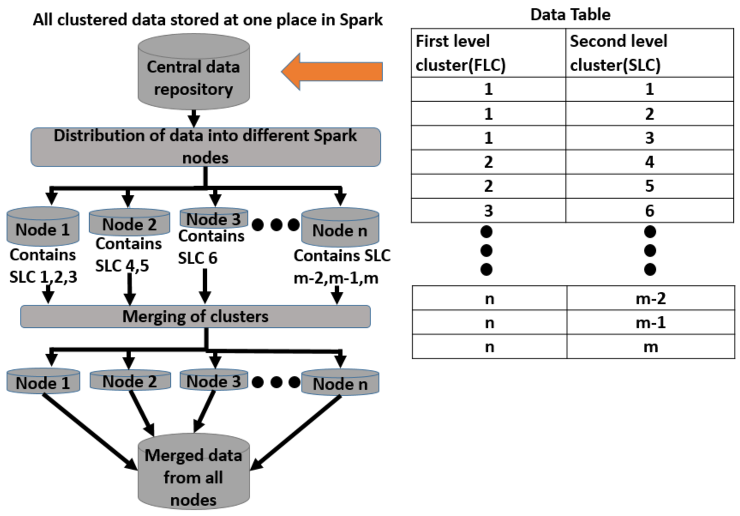
The approach for sequential and clique based merging for inter cluster merging is the same as that explained in intra level merging. Irrespective of what merging technique is employed, note that a major point of difference of our method from classical clustering techniques is that we specify the distance or similarity threshold for merging. This threshold automatically decides the optimal number of clusters to be made. In classical clustering methods, the big open question is to decide number of clusters, which is often done using some heuristic on evaluation metric values for different number of clusters. There are some method that automatically lead to a certain number of clusters, but it is not decided by an SME provided threshold.

3.3 Leveraging Apache Spark capabilities to parallelize each clustering algorithm:

Both sequential merging approach and clique-based approach involve a lot of computations. Both methods require calculation of distances between several pairs of second level clusters. Clique based merging approach involves finding maximum sized cliques with minimum average edge length. Searching for maximum sized cliques in a graph is a computationally expensive task and hence to carry out fast computations it is important to parallelize the algorithm as much as possible. We use Apache Spark, with custom defined map reduce operations to speed up the computations. In this section, we describe our use of Spark to do intra and inter clustering. First, the entire dataset is partitioned onto different nodes in Apache Spark, so that one node contains all the data for one given cluster of the first level. That is, if there are n first level clusters then there will be n nodes created on the Spark environment. Each of these nodes contain data of all the second level clusters pertaining to a specific first level cluster. The data is stored in the form of a Resilient Distributed Database (RDD) and the distance matrices computed are stored in the form of a matrix within each node. Merging of second level clusters pertaining to a specific first level cluster is done on the specific node in which the second level clusters are stored. The operations are executed parallely on each of the nodes so that second level clusters are merged parallely. Spark's capabilities with parallelism make the entire process much faster. The entire process is explained in Figure 5. For inter-level merging such distribution of data on Spark nodes is not effective as in intra level clustering because second level clusters are compared for merging across all the first level clusters. In this case we allowed Spark to intrinsically divide the data into 4 nodes and runs the algorithm parallely on the 4 nodes. Such parallelism was also helpful for executing the inter level merging.

3.4 Cluster evaluation

When comparing clusters generated by different methods results, the evaluation metrics break down the available information to a single number. These single number metrics are not easily interpretable and different metrics can give different ordering among the same clustering schemes being prepared. Some metrics might even give misleading information about the goodness of clusters formed. For example, spectral clustering can identify clusters of points distributed radially. If silhouette or Dunn



First level cluster(FLC)	Second level cluster(SLC)
1	1
1	2
1	3
2	4
2	5
3	6
●	●
●	●
●	●
n	m-2
n	m-1
n	m

Figure 4: Intra merging process using Spark capabilities.

index or Davies - Bouldin measure is used for measuring strength of such clusters, the results will seem very bad. Moreover, the standard metrics take all columns as inputs, without any notion of importance among them.

Here, we provide a method and describe its implementation to help evaluate clusters given by various methods in a holistic way. Instead of generating a single number, we generate a short but comprehensive report that business users can refer to while evaluating clustering quality.

Since we use many different clustering mechanisms over the same data with each mechanism involving only some of the columns provided, we need to establish a parity for comparing the resulting clustering schemes generated. As explained elsewhere in the paper, we take user inputs on which columns to cluster on and what taxonomy to use, it follows that cluster evaluation should follow the same methodology. That is, we take as inputs the columns considered as important (from clustering evaluation point of view) by business. In addition, we also take relative order/weights of importance of the columns. With this information provided, various clustering schemes involving different columns can be compared on the same scale. For example, for the job clustering case considered in this paper, one clustering scheme might use only job level and salary offered with K means, and another approach may use topic modeling based clustering on the provided job descriptions. Cluster quality evaluation on job level and salary only will, most probably, show the first scheme as better. And evaluation based on discipline and business group might show the other one as better.

Some of the attributes over which cluster quality evaluation (CQU) is to be done could be numeric, others could be qualitative. Given the context, out interest primarily lies in heterogeneity of each cluster. There could be several measures of heterogeneity measure of of a cluster over an attribute

Attribute type	heterogeneity measure	Range
Qualitative	Entropy	≥ 0
Qualitative	Variety index*	≥ 1
Qualitative	Mode Fraction	$(0, 1]$
Qualitative	Number of Unique values	≥ 1
Numeric	range	NA
Numeric	Interquartile Range	NA
Numeric	Standard Deviation	≥ 0
Numeric	Coefficient of variation	free

Variety index is a custom defined measure for spread of qualitative attributes. Let a qualitative attribute have values $v_1, v_2, ..., v_k$ in fractions $f_1, f_2, ..., f_k$. WLOG we can assume that the f_is are in decreasing order. Then the variety index of the attribute is

$$V = \sum_{i=1}^{k} i \times f_i$$

Note that if there is a tie in two f values, it can be broken arbitrarily. If there is only one value of the attribute, the variety index would be 1. If there is any variety, then it will be more than 1. Given its definition, the variety index penalizes having more values with increasing weightage. Rest of the heterogeneity measures explained above are standard measures from the literature. For numeric attributes, coefficient of variation (CV) is the best one since it is unitless and indifferent to the scale of attribute. CV may not be a good measure of dispersion if the mean is close to zero, because then it becomes very sensitive to changes in mean. In our case, since the numeric attributes are all reasonably large positive numbers, this issue does not exist. CV can also be effected by presence of outliers, hence in our implementation we provide a option of removing values that are outside a pre specified range. For the qualitative variables, Entropy and variety index are the most meaningful ones as they capture both mode fraction and number of unique values simultaneously. The reporting of the mode fraction and number of unique values might be useful from business user's perspective though.

Our system allows user to define which heterogeneity measure they want to use for cluster quality evaluation. We evaluate the chosen measures over all the clusters, and report the scores over attributes of interest. We also report the cluster size, number of unique values and an overall score (weighted by user importance) for each cluster. Eventually, the scores of all clusters formed within a clustering scheme are aggregated by weighing them with a user defined column (cluster size by default).

Since most of these heterogeneity measures are unbounded by definition. For both numeric and qualitative measures, lower values are better(correspond to low spread). But coefficient of variation and qualitative measures can be on different scales. In order to combine them meaningfully, some standardization needs to be superimposed. Moreover, from business user perspective, measures of spread being bounded is desirable. In order to achieve these goals, we normalize the measures on a 1 - 5 scale with 5 being the best possible value. Note that if we normalize each clustering scheme individually, it will be impossible to compare them. In order for the comparison to hold, the normalization of measures is done across all clustering schemes simultaneously. For example if we compare 2 clustering schemes, with c_1 and c_2 number of clusters, then normalization of the spread measures across various attributes is done on all the $c_1 + c_2$ values together.

4. Example analysis and results

The entire clustering methodology with both the approaches - sequential merging approach and clique based approach are tried out on real data sets. The dataset consisted of variables such as discipline, position title, salary range, career bands, and bonus plan pertaining to each job. initial task was to cluster different jobs based on different disciplines and position titles. Clustering was done using SME suggested keywords for disciplines and position titles by using string matching algorithm as described in 3.1. As mentioned in section 3.1, the threshold used for $C_{\max(|\mathbf{w}^i \in \mathcal{K}_{v_1}|, |\mathbf{w}^{i+1} \in \mathcal{K}_{v_1}|)}$ is set as,

$$C_n = \begin{cases} 1, & \text{if } n \leq 2 \\ \frac{2}{3}, & \text{if } n = 3 \\ \frac{1}{2}, & \text{if } n = 4 \end{cases} \tag{4}$$

For this problem, there were at most 4 keywords in discipline or position title. So, $n > 4$ is not considered in the equation (4). In total there were 247000 data instances and those were clustered in 16 first level clusters based on job discipline and 512 second level clusters based on position title. Essentially 512 clusters is too huge a number and hence it was required to merge second level clusters based on salary range and career bands. For this we tried both the approaches

– Sequential merging approach

– Clique based approach.

as described in 3.2. For the clique based approach the distance threshold used is 0.15. To keep parity with clique based approach, in the sequential merging approach the similarity score threshold used was $1 - 0.15 = 0.85$. A sequential merging approach was also executed on the usual 8 core machine (without spark capabilities and parallelization) to compare the the results with the same obtained using Spark capabilities through parallelization. Due to expensive computations the clique based approach was implemented only with spark capabilities through parallelization. For merging, all three approaches were divided in two parts, intra level merging, inter level merging as explained in 3.2. Using sequential merging approach, through intra level merging 16 first level clusters and 191 second level clusters were created. For inter merging of clusters, the input was that obtained from intra level merging. Again, using sequential merging approach, through inter and intra level merging finally 13 first level clusters and 60 second level clusters were obtained. As expected for sequential approach results obtained through parallelization and without parallelization were the same. Using clique based approach, through intra level merging 16 first level clusters and 190 second level clusters were created. As mentioned before for inter merging of clusters, the input was that obtained from intra level merging. Again, using clique based approach, through inter level merging finally 13 first level clusters and 60 second level clusters were obtained. To record the average time taken to run intra level merging and inter level merging, the algorithms were re-run 100 times on the same data and runtime was recorded. For intra level merging the average recorded timings are as shown in Figure 5. For inter level merging the average run time is shown in Figure 6 The average time recorded for all three methods is shown

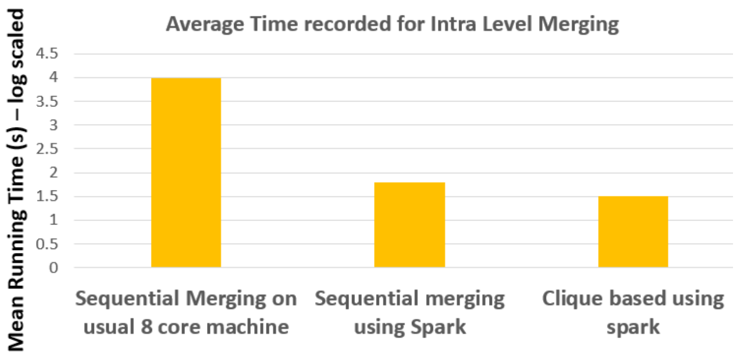


Figure 5: Average recorded timings for intra merging.

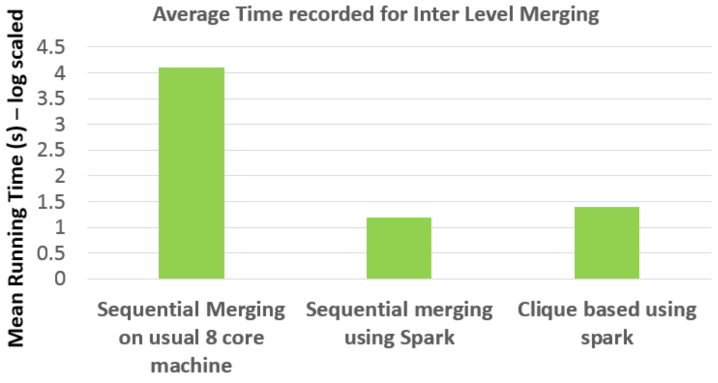


Figure 6: Average recorded timings for inter merging.

in Figure 7. From the figures it can be seen that parallel implementations leveraging Spark capabilities perform appreciably faster in time as compared to the usual one executed on a 8-core machine all the while preserving the quality of cluster after merging. Inter level merging in clique based approach takes less time as compared to linear merging. This result might seem counter intuitive but a detailed investigation into data reveals that the graph obtained for our data is highly sparse. Among all the edges only 4% of the edges are less than the distance threshold 0.15 and hence makes the entire graph highly sparse.Thereby reducing the number of cliques sufficiently. Since the number of dense subgraphs are rare. This algorithm performs faster in our case. Please note that the clique finding algorithms can grow exponentially because adding one node into the graph may lead to exponential rise in the number of cliques. But, since all the maximal cliques are computed in one iteration and after that we check if the new node formed after merging of the clusters makes clique with some already existing clique. Thus, if the number of cliques in first iteration were not too large, the algorithm therefore will perform good in terms of time. But one must be warned at this moment that given a highly dense and compact data set or data set wherein clusters are not sufficiently isolated, the trend in this algorithm may not generalize well. Also, from Figure 5 it can be seen that in intra level cluster merging, graph is still sparse and there are not many cliques in each first level clusters. But since the cluster searching in the first iteration requires a longer period of time, which is of exponential order to the number of nodes in the graph, the clique based merging takes more time in searching the clique. Sequential merge however avoids this computation intensive task and hence performs well.

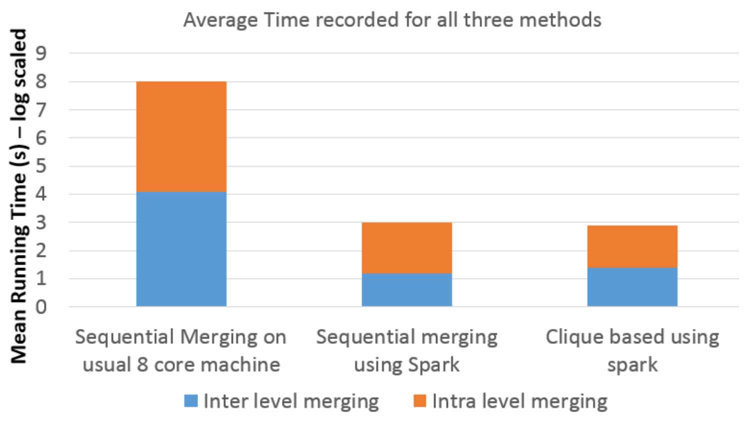


Figure 7: Average recorded timings for all three approached.

In nutshell, it can be said that in large but sparse subgraphs the clique based merging would produce faster results. But if the graph is dense the clique based approach would be a bad algorithm in terms of time. Also, in sufficiently small graphs, the sequential merging would produce fast results. Apart from time taken by these algorithms in producing results, another important facet of the result is the analysis of quality of merged clusters obtained from each algorithm. One would note that the cliques are very strong communities and are tightly packed. Due to stepwise concatenation, the sequential merge may produce a cluster containing two sufficiently dissimilar clusters, but clique based merging would not let it happen. The quality of the merged clusters in clique based merging is guaranteed to be good. While the sequential merging can produce the good results if the clusters are compact and isolated. An analysis of the quality of clusters obtained is given below.

When cluster quality evaluation is done on clusters created by both sequential and clique based methods, the overall numbers obtained are 2.303 and 2.309 respectively (out of maximum possible 5). But when the detailed report is analyzed, we find that 40 clusters in both the methods are of size less than 10 unique position titles (figure 8). These smaller clusters have the best spread numbers in both numeric and qualitative variables. Since the overall cluster evaluation measure is normalized on the basis of maximum and minimum values, these smaller clusters tend to skew them.

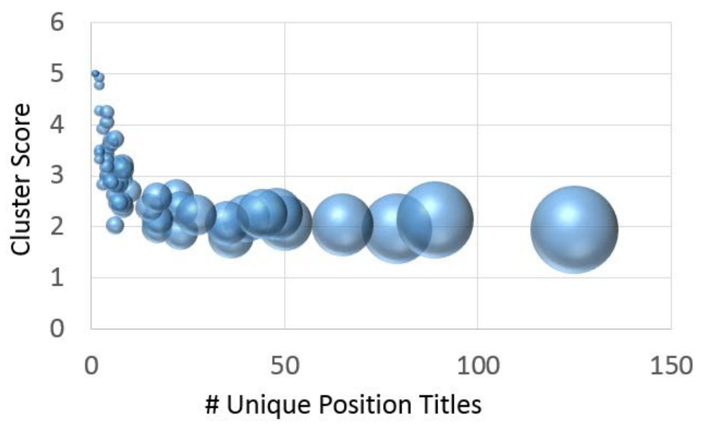


Figure 8: score vs unique position titles. Bubble size $\propto$ number of entries.

As can be seen, scores among the larger clusters are in a relatively small range. Similar pattern is observed for clique based merging as well. SMEs validated that results from both the approaches were in sync with their understanding.

5. A Case Study

Now we present a case study on a real data set where we come up with different benchmarks for number of days taken to close similar job requisitions corresponding to each cluster. Also, to further analyze the delay in different stages of recruitment process we come up with benchmarks for

– Number of days to close a requisition

– Number of days between approval of requisition to opening to the requisition

– Number of days between creation of requisition to approval of the requisition

– Proportion of applications received for a requisition to screened

– Proportion of candidates called for interview to screened

– Proportion of candidates rejected in the interview to number of candidates called for the interview

– Proportion of candidates rejected an offer to the number of candidates to which offers were extended.

  The analysis is done on a real recruitment data coming from a US based industry. The data set consisted of job requisitions and the corresponding job discipline, position title, salary range (high salary amount, low salary amount), career bands, bonus plans. The number of days taken to close that requisition, the number of days taken to approval of job requisition to opening, the number of days taken to create the job requisition to approval of requisition are also provided. In total there were 247000 data instances. There could be many requisitions which appeared multiple times in the data as many candidates applied against that requisition. For such cases, all the variables pertaining to the requisition have the same entry but the entries associated with candidate data change.

  Using SME insight and by customized clustering, all the job requisitions were clustered into 33 clusters. Next an analysis on the number of days taken to close (DTC) the requisition was done. As it can be seen in Table 1 cluster number 24 has 432 distinct job requisitions closed in the entire of two years. Within that cluster there are requisitions which got closed in 0 days (i.e. on the same day it opened) while the median number of days taken to close the requisitions in cluster 24 is 24 days. We set median number of days to be the benchmark for a cluster since, unlike mean, it is not sensitive to outliers. So, for cluster 24, requisitions which got closed before 24 days are meeting the benchmark while the rest of the requisitions are not meeting the benchmark and hence having a longer recruiting process. Next we considered only the requisitions that are closed in the recent 3 months computed from the last closing day of a requisition. For each cluster, we determine, among requisitions that got closed in the last 90 days, how many of those did not meet the

Table 1: Analysis of number of days to close for top five clusters containing most distinct requisitions

Cluster Number	Number of Requisitions	DTC min	DTC Q1	DTC median	DTC Q3
24	432	0	11.00	24.0	45.00
6	423	5	41.00	80.0	126.50
32	184	12	48.00	75.5	123.75
7	150	4	40.25	72.5	110.00
33	128	8	53.00	85.0	121.00

Table 2: Number of requisitions under-performing in each cluster.

Cluster Number	Number of Underperforming Requisitions
6	30
24	25
33	8
26	7
7	7

days to close benchmark that was calculated in Table 1 based on all the requisitions over the whole time period of two years for each cluster. It can be seen from Table 2 that maximum number of under-performing requisitions are in cluster 6. There are 30 requisitions which are under-performing. The top 5 clusters with maximum number of requisitions under-performing are shown in Table 2. In total 248 requisitions were closed in the last 3 months and 142 of those requisitions under-performed and did not meet the cluster wise benchmark value. Further, for the requisitions closed in the last 3 months, we wanted to check which States are having most number of under performing requisitions. From Figure 9 it can be seen that the maximum number of requisitions that are under-performing are in Michigan. Also a huge number of requisitions are under-performing in Florida followed by Texas, Tennessee and Massachusetts. In order to capture the cluster wise under performance across all the states we created the heatmap as shown in Figure 10. The heat map shows severity of under performance within each state, for every cluster. The coloring scheme of the heatmap is based on the following rule:

∗ If there are more than 14 requisitions within a cluster-State combination and more than 50% of those requisitions are under-performing then the heat map color is red for that cluster-State combination.

∗ If there are less than 14 requisitions within a cluster-State combination and more than 50% of those requisitions are under-performing then the heat map color is coral red for that cluster-State combination.

∗ If there are more than 14 requisitions within a cluster-State combination and less than 50% of those requisitions are under-performing then the heat map color is dark green for that cluster-State combination.

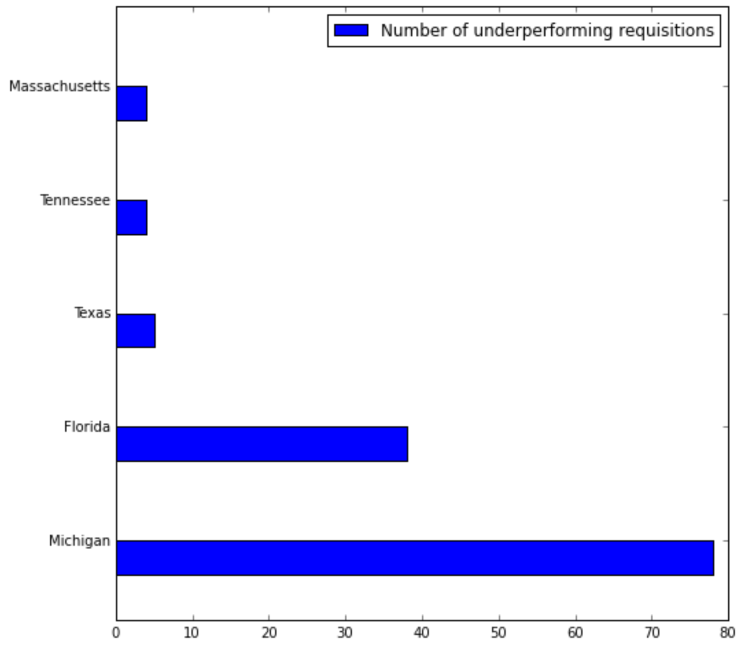


Figure 9: States with most number of under-performing requisitions.

* If there are less than 14 requisitions within a cluster-State combination and less than 50% of those requisitions are under-performing then the heat map color is light green for that cluster-State combination.
* If there are no requisitions in a cluster-State combination then the color is blue.

As it can be seen from the heatmap that most of the requisitions are under-performing in Michigan across all clusters. Also in Florida, a substantial number of requisitions are under-performing across all clusters. In California all the requisitions were closed within the specified benchmarks across different clusters. Looking at the heatmap it can be concluded that Michigan is the worst performing State across all the clusters in terms of closing requisitions within the specified benchmark. In order to understand why the requisitions are under-performing in Michigan following benchmarks are calculated for each cluster.

* Benchmark 1: Proportion of applications reviewed for that job requisition to total number of applications received for a job requisition.
* Benchmark 2: Proportion of applicants called for interviews to number of applications screened
* Benchmark 3: Proportion of candidates rejected after interview to number applicants called for interview
* Benchmark 4: Proportion of candidates rejected the offer to the number of candidates to whom the job was offered
* Benchmark 5: Benchmark for number of days between requisition created to requisition approved
* Benchmark 6: Benchmark for number of days between requisition approved to requisition open

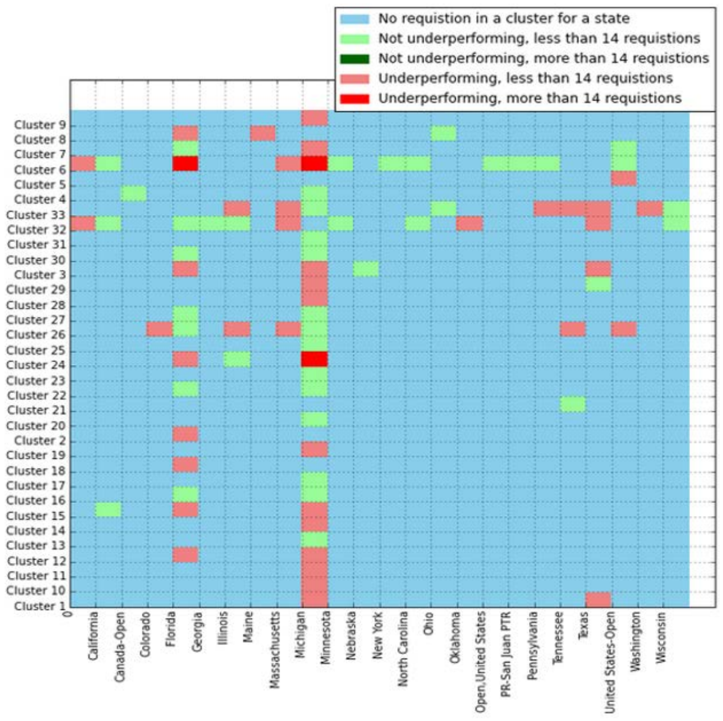


Figure 10: Heatmap of rate of under performance of requisitions in over all States within each cluster .

∗ Benchmark 7: Benchmark for how many days it took for the entire process (requisition created to requisition closed)

For each cluster, median value was taken as the benchmark threshold derived from candidate data or requisition data within that cluster. After calculating the benchmarks it was analyzed which benchmarks are being violated in the State Michigan across most of the clusters. The heatmap shows the performance of each of the benchmarks across all the clusters for the State of Michigan.

∗ Across most of the clusters, performance in the state of Michigan is poor for all the benchmarks except benchmark 4.
∗ Michigan's performance for benchmark 2 is poor across most of the clusters. That is, many candidates applying in Michigan are rejected in screening and very few are interviewed.
∗ Performance of Michigan on benchmark 1 is poor across most of the clusters. That means in Michigan many job applications are not reviewed.
∗ On benchmark 3 as well, the performance of Michigan is poor across some of the clusters. That means in Michigan many candidates are rejected in interview also.
∗ In Addition, Michigan's performance on benchmarks 5, 6 and 7 are also poor across clusters.

It can be recommended that to perform better, the ratio of number of applicants to those passing the screening stage needs to be improved in Michigan. Job applications are also needed to be reviewed pro-actively. Perhaps due to a huge number of requisitions being created in Michigan, the entire process of requisition approval, opening time are getting delayed.

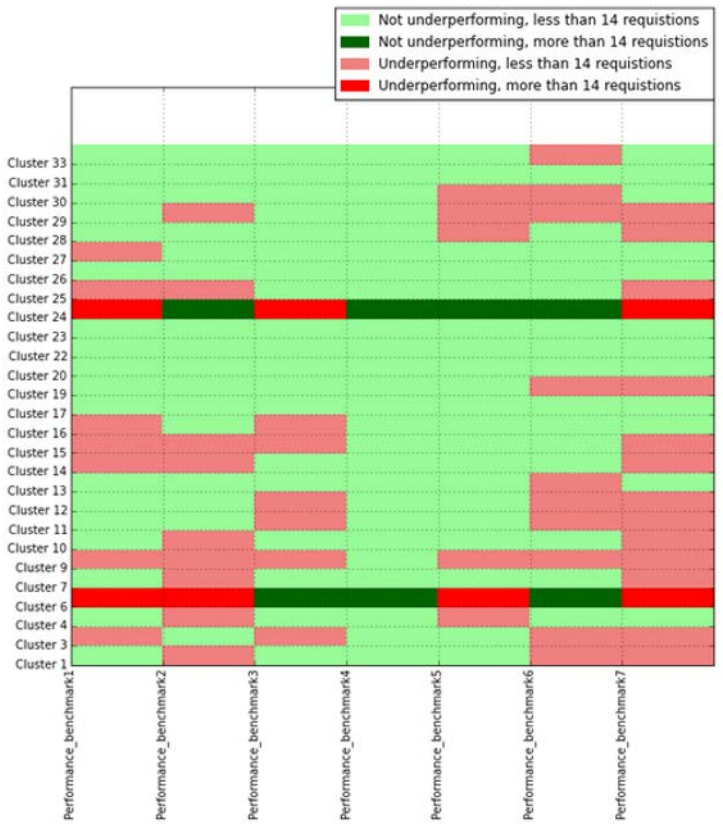


Figure 11: Heatmap of performance on various benchmarks across all the clusters for the State Michigan.

6. Conclusion and Future Directions

Clustering records to create reliable benchmarks is an approach that researchers have been taking in recent past. In this paper, we create a customized clustering algorithm to cluster similar jobs (described by position title, salary band, discipline, business group etc) together. Our method involves grouping and re-grouping jobs based on SME input for variables to group on, and their importance order. The method also takes as an input a similarity threshold to combine clusters formed at intermediate stages. This threshold causes the number of clusters to converge to an optimal value by itself. Hence the user needs to decide an appropriate threshold for this method, as opposed to number of clusters which is required for most traditional clustering techniques.

In addition to creating the clusters, business users often need a report to check clustering quality since a single number metric doe not provide the necessary details. Our proposed report takes the important attributes to evaluate the clustering scheme over as input, and generates a detailed report with scores of spread among the clusters over each individual attribute. An overall normalized clustering quality score is generated for SMEs to get a high level idea of which clustering scheme is doing better in general. The overall score might get skewed to a small number due to smaller clusters having much more homogeneous data than larger ones. The detailed report can help the user identify such an occurrence and help in rectifying it.

We validated our results and reports with SMEs, and this approach gives more business appropriate results than traditional clustering schemes. The next steps are to devise a method to overcome biasing of score due to smaller clusters. Also, some clusters formed are very small in size. We can change the cluster merging criteria to be controlled by both distance threshold and relative size.

References

[1] S. Kleinhans, C. Merle and G. Doumeingts, *Determination of what to benchmark: a customer-oriented methodology.* Benchmarking Theory and Practice, pages 267–276

[2] A.K. Jain, M.N. Murty and P.J. Flynn, *Data clustering: a review.*, ACM computing surveys (CSUR) 31, no. 3 (1999): pages 264–323.

[3] A.K. Jai and, R.C. Dubes, *Algorithms for clustering data.*, Prentice-Hall, Inc., 1988.

[4] X. Dai and T. Kuosmanen, *Best–practice benchmarking using clustering methods: Application to energy regulation.* Omega, Volume 42, Issue 1, January 2014, pages 179-188

[5] M.J. Sharma and S.J. Yu, *Performance based stratification and clustering for benchmarking of container terminals*, Expert Systems with Applications, Volume 36, Issue 3, Part 1, April 2009, pages 5016-5022

[6] D.L. Davies and D.W. Bouldin, *A Cluster Separation Measure*, IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-1 (2), 1979. pages 224-227

[7] J.C. Dunn, *A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters*, Journal of Cybernetics 3 (3), 1973. pages 32-57

[8] P.J. Rousseeuw, *Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis*, Computational and Applied Mathematics 20, 1987. pages 53-65

[9] E. Achtert, S. Goldhofer, H.P. Kriegel, E. Schubert, A. Zimek *Evaluation of Clusterings Metrics and Visual Support.* 2012 IEEE 28th International Conference on Data Engineering. pages 1285–1288

[10] M. Meil *Comparing clusterings an axiomatic view.* Proceedings of the 22nd international conference on Machine learning, 2005, pages 577–584