SURF

Entity Matching and Resolution System for Indian Names by TCPD (Ashoka University)

By Aditya Garg (IIITD)

Table of Contents: -

1. Introduction: -

SURF is an entity resolution tool built by TCPD for Indian names. The tool is designed to resolve names of candidates in order to create a unique identifier for each candidate that
has ever contested an election. It helps in identifying multiple entries of the same person with little or no variation in his/her information thus reducing the duplicity in the records.

SURF clusters record based on the resolution variable i.e. Candidate Name in this context, based on some clustering algorithm like edit distance, cosine similarity, compatible names etc. The user has the option to choose from a given set of predefined algorithms. The user also has the facility to add clusters for review which can be later reviewed by an analyst. The user also has the option to filter clusters by parameters like Age, Name, etc. After SURF clustering and reviewing, a human analyst merges or unmerges records based on his/her knowledge. The user can then download the final record with all the implemented changes.

2. Technical Background: -

SURF takes in a dataset consisting of information about political candidates who had contested elections. The tool then tokenizes the information row wise (like Mr. Aditya Garg is tokenized to [Mr., Aditya, Garg]) and collects the relevant information which is then sent for canonicalization (Dr. Harsh Kumar Vardhan is canonicalized to HK Vardhan) thus removing honorifics, irrelevant middle/last names, etc. The canonicalized data is then processed by the algorithm (edit distance, cosine similarity, compatible names, etc.) which is chosen by the user. The algorithm clusters the rows accordingly and shows the user the result. SURF's user interface provides certain operations, such as searching for a particular person, reviewing a cluster for further consideration by an analyst, merging and unmerging rows to identify them as a single entity, filtering clusters according to some parameter, etc. Tools and Technologies Used: Java, Java Servlets, JSP, HTML, CSS, JavaScript, Apache Maven, Apache Tomcat v9.0, Apache Ant, etc.

3. My Work: -

I developed two algorithms i.e. the Cosine similarity algorithm and the Review algorithm for the SURF. I also edited the edit distance algorithm, compatible names algorithm in SURF. The algorithms are as explained:

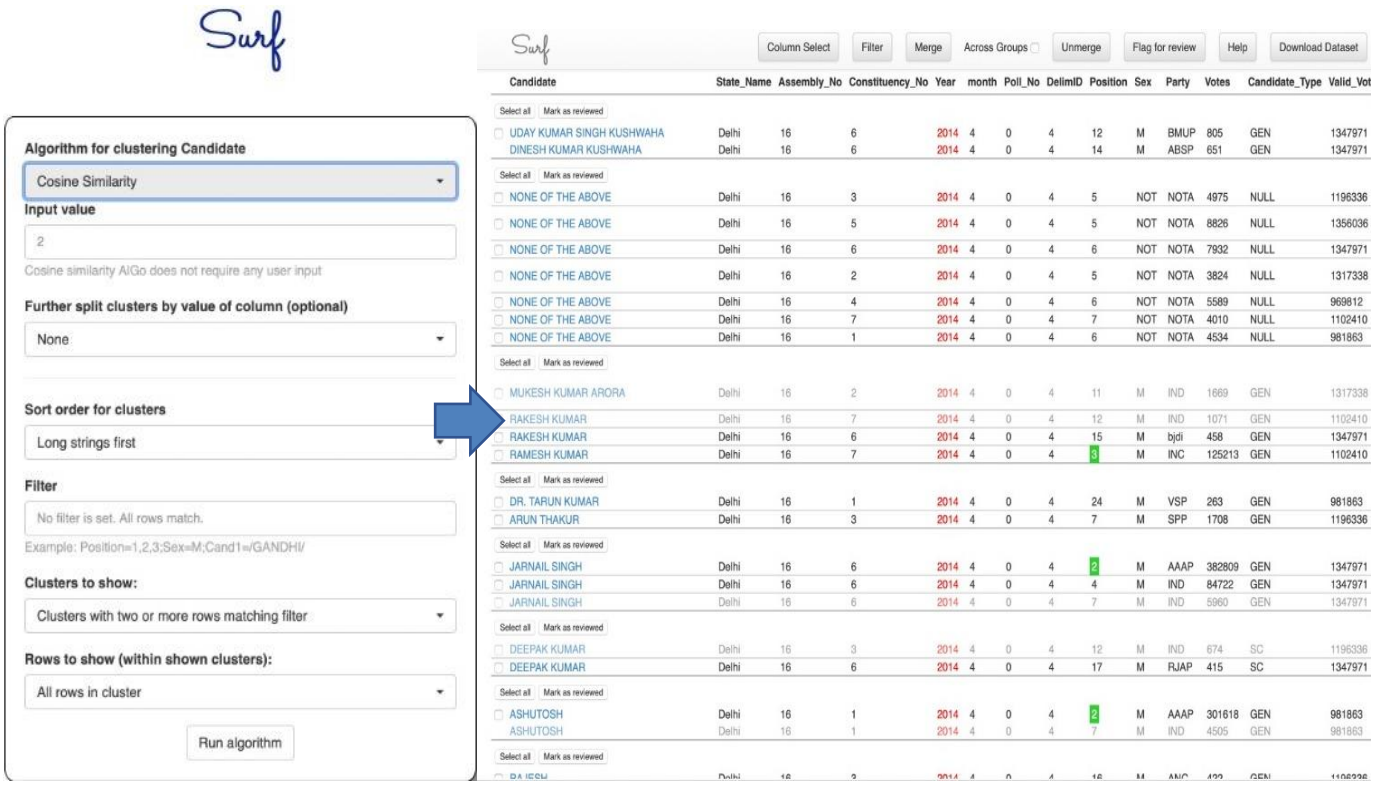
3.1 Cosine Similarity Clustering Algorithm

The Cosine similarity algorithm works on the principle of cosine similarity between two strings in which the strings are considered as vectors and the similarity between them is found using the cosine formula. The algorithm converts the resolution variable (i.e. Candidate Name in this context) to a vector of at-max 26 dimensions representing the 26 alphabets, Example: Aditya is converted to 2a + 1d + 1i + 1t + 1y, Naman to 2n + 2a + 1m, etc.
The computed vectors are then used to find the similarity between two entries by judging their cosΘ value in the computed cosine formula.

The Formula: $$cos\Theta = \left| \frac{(Vector\ of\ String\ 1).(Vector\ of\ string\ 2)}{(length\ of\ Vector\ 1) * (length\ of\ Vector\ 2)} \right|$$

A value of $cos\Theta = 1$ indicates perfect similarity between the two entries and $cos\Theta = 0$ indicates perfect dis-similarity between the two entries. The user is prompted to provide "InputVal" which indicates perfection between the entries in the cluster.
Example: InputVal = 95 indicates that the clusters will have entries with $cos\Theta \geq 0.95$.

The user can then merge/unmerge the entries as per his knowledge. The user can also flag clusters for review which can be later merged/unmerged by an analyst. The working depiction is as follows: -



Cosine Similarity Chosen (Left) and Cosine Similarity Result (On Delhi Assembly Elections-2019 with InputVal = 95) (Right)

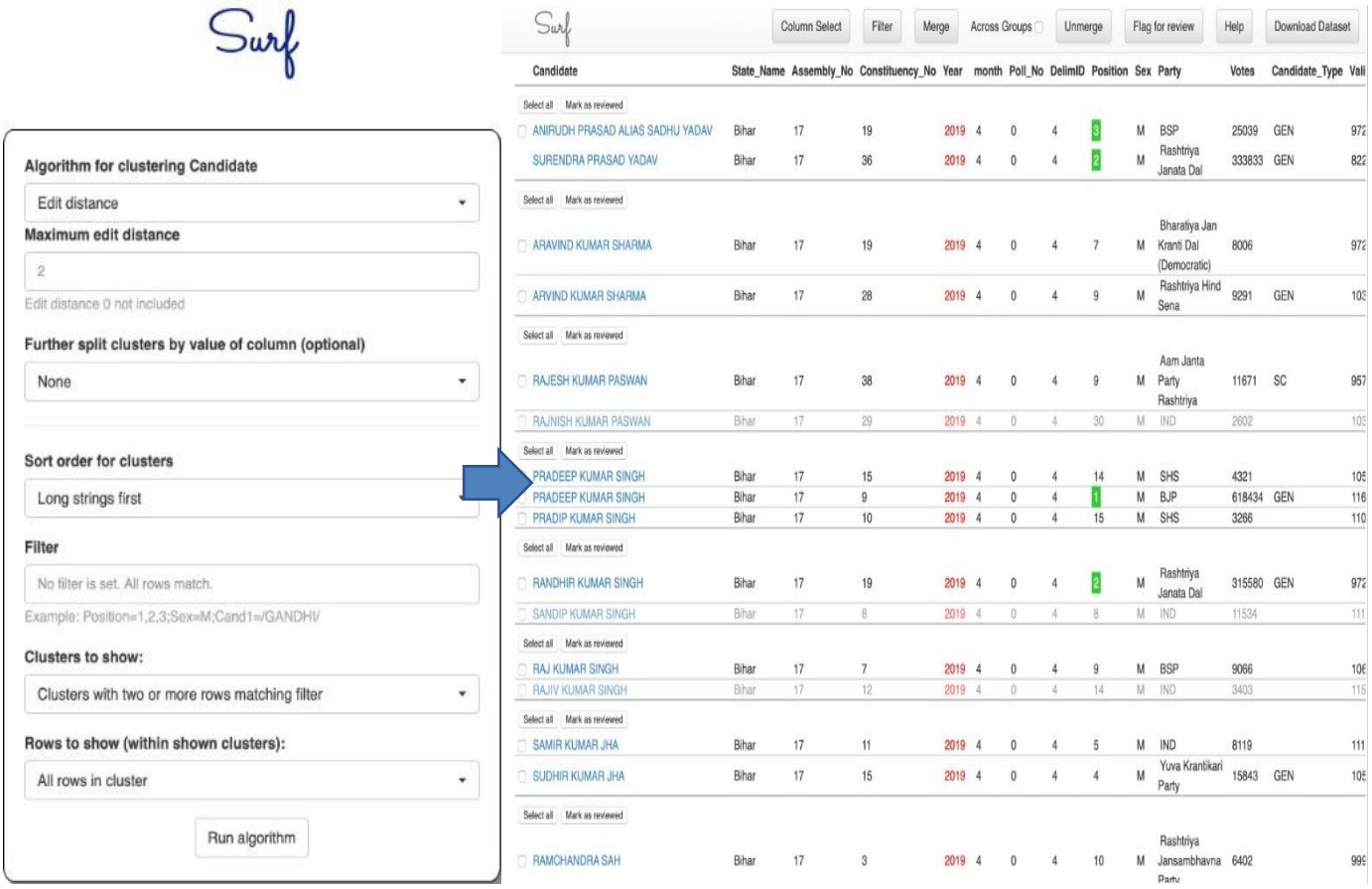
3.2 Edit Distance Clustering Algorithm

The edit distance clustering algorithm works on the principle of edit distance algorithm (Levenshtein distance algorithm) in which the similarity between the two strings is measured based on the number of deletions, insertion, substitution of the alphabets in one string to make it exactly similar to another. The algorithms make use of dynamic programming to store results of its subproblems for faster processing and use multi-threading for the same purpose. (Source: Tsinghua Group)

$No\ of\ edits\ =\ 0$ indicates that the two strings are identical.
$No\ of\ edits\ \geq\ Min(len(String\ 1), len(String\ 2))$ indicates that the two strings are completely dissimilar.

The user is prompted to provide a "Maximum edit distance" which indicates the maximum number of edits the algorithm can do to a string to make it identical to the second one, a cluster will have entries that have Number of Edits <= "Maximum edit distance" themselves. Example: Maximum edit distance = 2 indicates that the clusters will have entries with $No.of\ edits \leq 2$

The user can then merge/unmerge the entries as per his knowledge. The user can also flag clusters for review which can be later merged/unmerged by an analyst. The working depiction is as follows: -



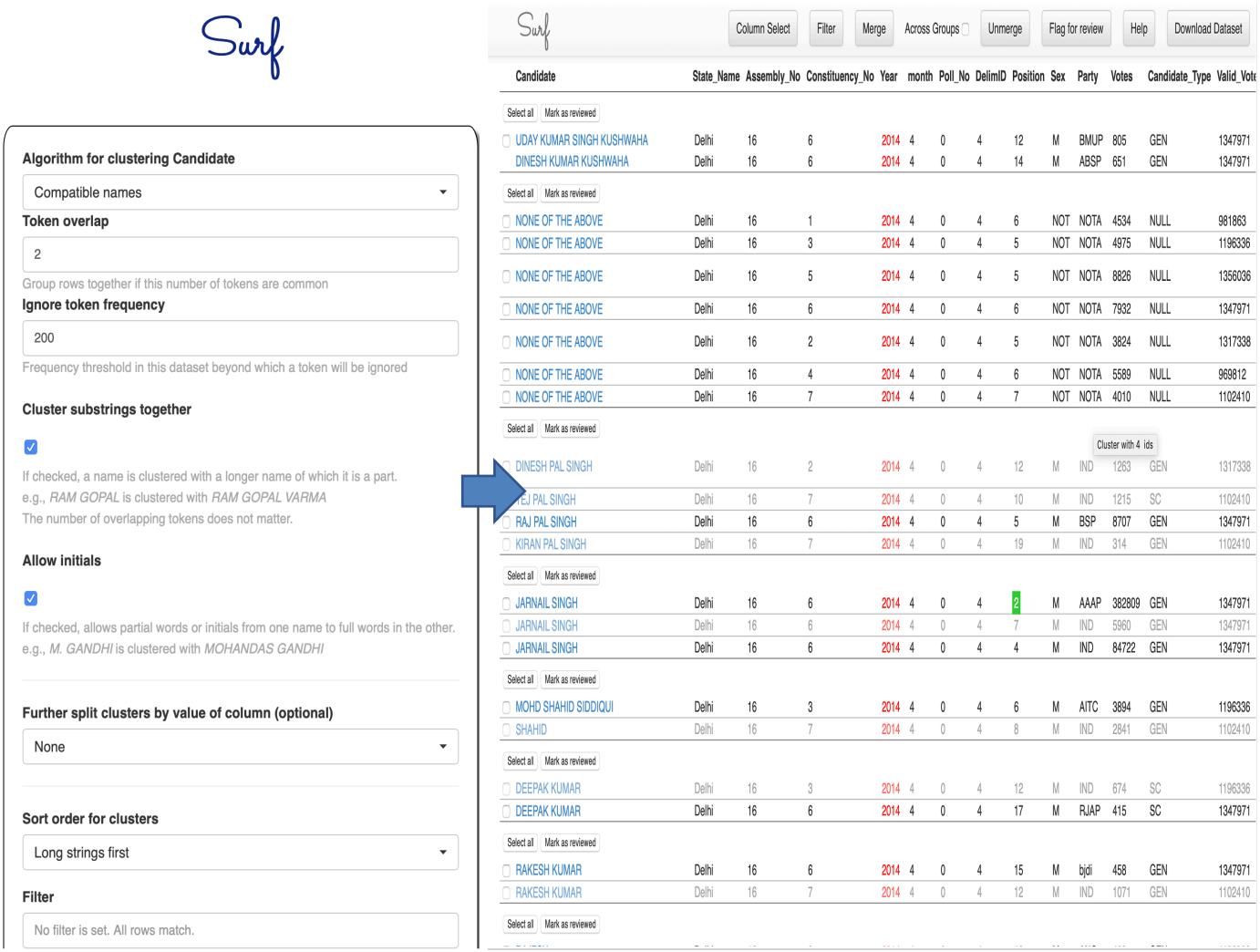
Edit Distance chosen (Left) and Edit Distance Result (On Bihar Assembly Elections-2015 with Maximum edit distance = 2) (Right)

3.3 Compatible Names Clustering Algorithm

The compatible name clustering algorithm works on the principle of number of token matching between two strings. In it each string is broken down into tokens (Example: Dr. Ram Kumar Garg is tokenized to [Dr.,Ram,Kumar,Garg]). After each string is tokenized, the algorithm compares the number of token overlaps between two strings. Normal token overlaps means the First alphabet of both the tokens are the same (Like: R and Ram overlaps). To compare two strings there should be one token which completely overlaps (i.e. Ram completely overlaps with Ram).
The user is prompted to provide a "Token overlap" which indicates the minimum token overlap between two strings to be in a cluster.

Example : Token overlap = 2 means that if two strings have token overlap (one complete and other normal) >= 2, they will belong to the same cluster.

If Token overlaps between two strings = 0, it indicates that the two strings are completely dissimilar.
If Token overlaps between two strings = MaxToken(String1,String2), it indicates that the two strings are identical.

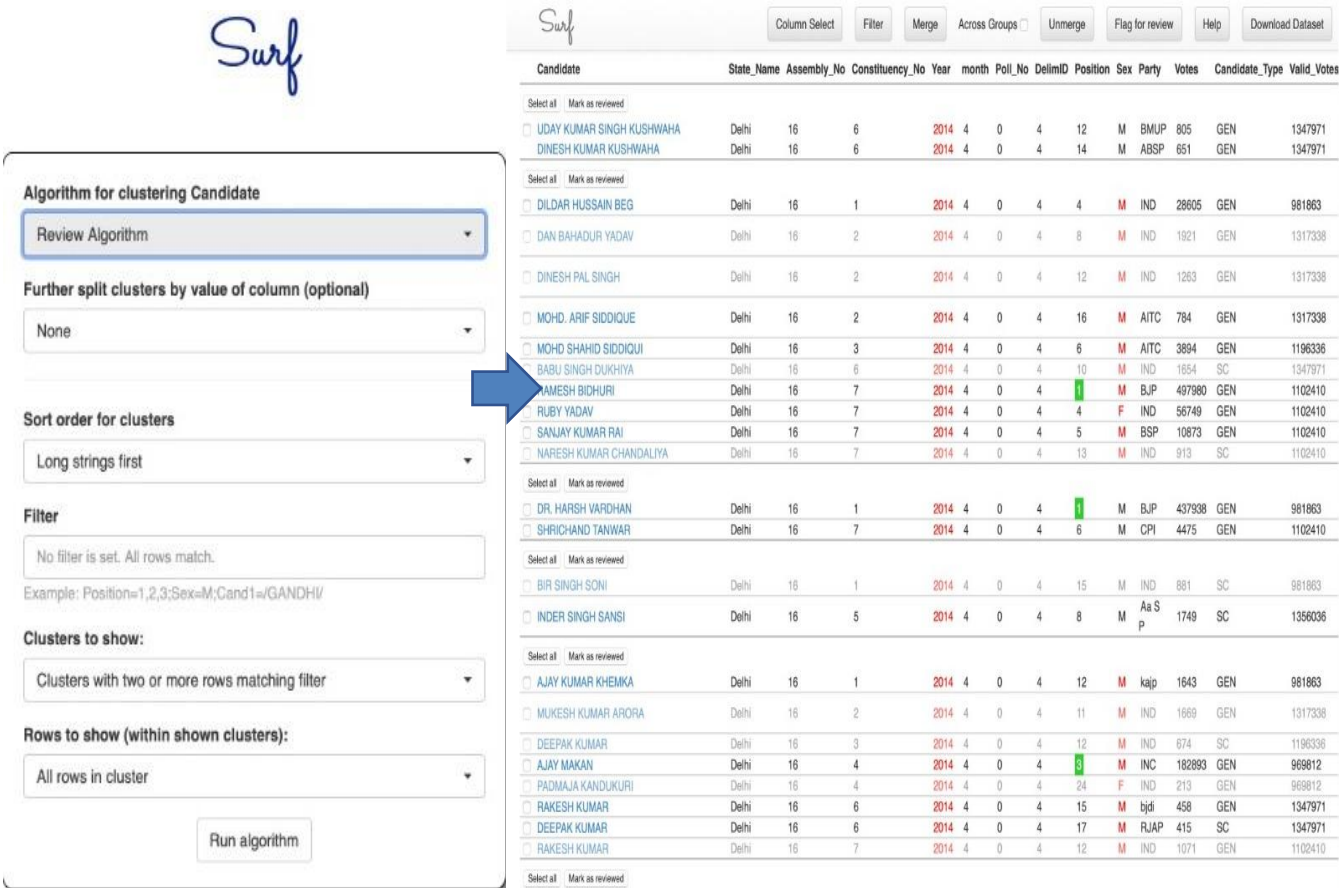


Compatible Name Algorithm chosen (Left) and Compatible Name Algorithm Result (On dataset of Delhi Assembly Elections-2019), all the clusters shown were flagged for review by the user (Right)

3.4 Review Algorithm

The review algorithm is used to show all the clusters that were flagged for review by the user for the given dataset. This algorithm is mainly used by an analyst to merge/unmerge entries in clusters according to his knowledge. Every file in SURF has a column named “Rid” initialized with 0, which is used to identify its review id. Entries with the same Rid are added to a single cluster and shown in the review algorithm for the analyst to review, if the cluster marked for review has some rows already marked for review before, it will combine those two clusters to a single cluster.
The Review algorithm saves the analyst’s time as he/she doesn't have to go over the whole document again for the purpose of merging/unmerging, Also. helps the user to visualize data using two different clustering algorithms.

The working depiction of the Review algorithm is as follows: -



Review Algorithm chosen (Left) and Review Algorithm Result (On dataset of Delhi Assembly Elections-2019), all the clusters shown were flagged for review by the user (Right)

3.5 Space and Time Complexity Analysis

The space and time complexity of the algorithms in SURF are as follows: -

1. Tokenization and Canonicalization: O(n) time and O(n) space complexity. This process runs before every algorithm.
2. Cosine Similarity Clustering Algorithm: It takes ~ $O(l^2n^2)$ time and ~ $O(n^2)$ space to compute.
3. Review Algorithm: It takes ~ O(n) time and O(n) space to compute.

Note: - Here n is the number of rows in the dataset and l is the maximum length of the resolution variable i.e. Candidate Name in this context.

The below table depicts the time complexity of SURF algorithms on some sample dataset taken from http://lokdhaba.ashoka.edu.in. It is as follows: -

S. No	Name of Dataset	No. of Rows	Edit Distance (Max=2)	Edit Distance (Max=4)	Cosine Similarity InputVal=95	Cosine Similarity InputVal=90	Review Algorithm	Compatible Name Algorithm	Tokenization Time
1.	Delhi	7199	~165ms	~315ms	~11.9s	~11.9s	<50ms	~1.07s	500ms
2.	Kerala	10820	~265ms	~490ms	~28.2s	~28.5s	<100ms	~2.073s	484ms
3.	Tripura	3001	~98ms	~85ms	~2.58s	~2.37s	<10ms	~1.56s	198ms
4.	Gujrat	17082	~415ms	~612ms	~41.83s	~41.12s	<150ms	~2.8s	967ms
5.	General Elections (2019)	8962	~258ms	~282ms	~20.59s	~20.63s	<100ms	~1.76s	580ms

Time Complexity

S. No	Name of Dataset	No. of Rows	Edit Distance (Max=2)	Edit Distance (Max=4)	Cosine Similarity InputVal=95	Cosine Similarity InputVal=90	Review Algorithm	Compatible Name Algorithm
1.	Delhi	7199	351	162	1234	997	-	330
2.	Kerala	10820	635	238	1761	1354	-	380
3.	Tripura	3001	245	170	545	381	-	281
4.	Gujrat	17082	811	513	2767	2234	-	587
5.	General Elections (2019)	8962	334	257	1142	1126	-	286

Number of Clusters Formed

Note: - 1. The time complexity was taken on a MacBook Pro Intel core i5. Time complexity with other processors may vary.
2. (*) The Time complexity for the first run of the Compatible Names algorithm is more as compared to its subsequent runs. The time taken for the first run is ~ 20-30s.

4. Conclusion

Thus SURF helps in resolving names of political candidates via creating unique identifiers for them. This in turn reduces the redundancy and confusion in data which can be used more efficiently for other analyses. Some of the main conclusions are :-

1. The User should use a combination of different algorithms to analyze the data more efficiently.
2. In merge algorithms, as we relaxes the similarity factor (i.e. “Maximum edit distance”, “InputVal” etc), the no of clusters decrease and the size of the clusters increases.
3. The relative run time of the merge algorithms are as follows:-

   Edit distance << Compatible Name < Cosine Similarity

Also, SURF provides the user the ability to analyze political data more efficiently by using filters parameters like Age, Name, Year etc. and by reviewing clusters which can be very useful for merging data with the help of a human analyst.

5. What’s Next

Although the SURF model helps in Entity Matching and Resolution in a very effective way the model can be improved. Some ideas to improve SURF are as follows :-

1. Currently the Cosine Similarity Clustering Algorithm doesn’t account for anagrams. This limitation can be removed by taking cosine similarity of n-grams of the string.
2. None of the Algorithms account for clustering similar sounding names like :- Joseph and Josef. A Soundex algorithm can be added to remove this limitation.
3. The time taken by the Compatible Name and Cosine Similarity Algorithm can be reduced by using multi-threading, currently they are single threaded algorithms.
4. The model can be improved by enabling the tool to run an algorithm on the output of an algorithm i.e. clustering within clustering. Currently a single algorithm runs at a time.
5. The model can further be improved by combining the results of different algorithms and show it as a single result to the user.
6. The model can be converted to a machine learning based tool, thus improving the accuracy and efficiency of the model.

6. How to run SURF

To run SURF, open the terminal/Command prompt and run the following commands after cloning the SURF repository from git repository. The commands to run are:

1. In ‘surf’ directory run **“mvn”**
2. Move to ‘surf-launcher’ directory
3. In ‘surf-launcher’ directory run **“mvn”**
4. Now run **“ant standalone-jar”**
5. Finally run **“java -jar surf-standalone.jar”**

To run SURF, Apache Maven (mvn) and Apache Ant (ant) dependencies are required. One can download the following from:

Apache Maven (mvn) : https://maven.apache.org/download.cgi
Apache Ant (ant) : https://ant.apache.org/bindownload.cgi

Recommended Versions :-

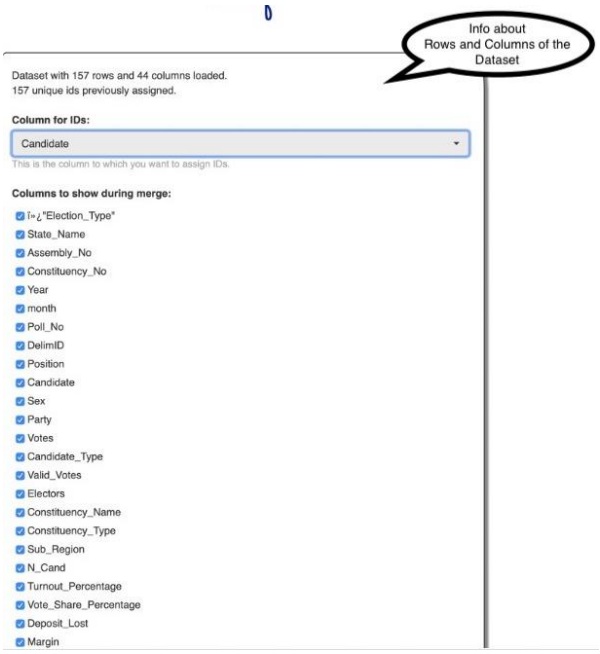
1. Recommended v3.6.3 tar.gz/zip file for Apache Maven
2. Recommended v1.10.8 tar.gz/zip file for Apache Ant

7. How to run SURF user interface

A person can run the SURF user interface as follows: -

The Starting Window

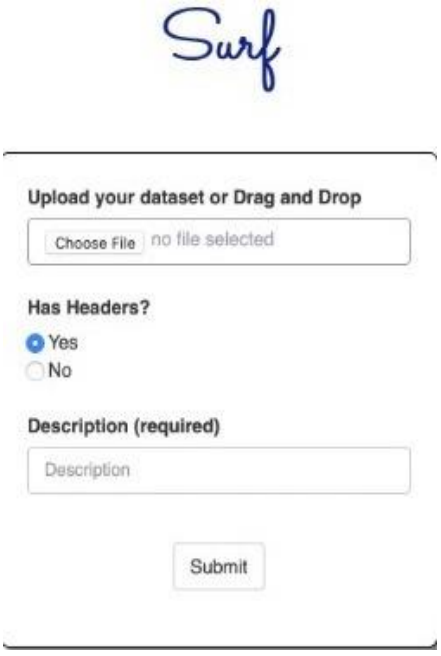
Choose Dataset & Click Load Dataset



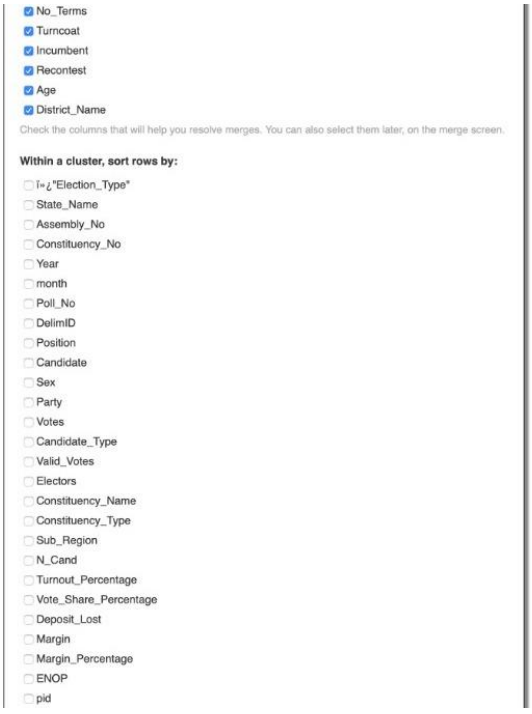
Choose the Resolution Variable

Upload Dataset

Choose Filter

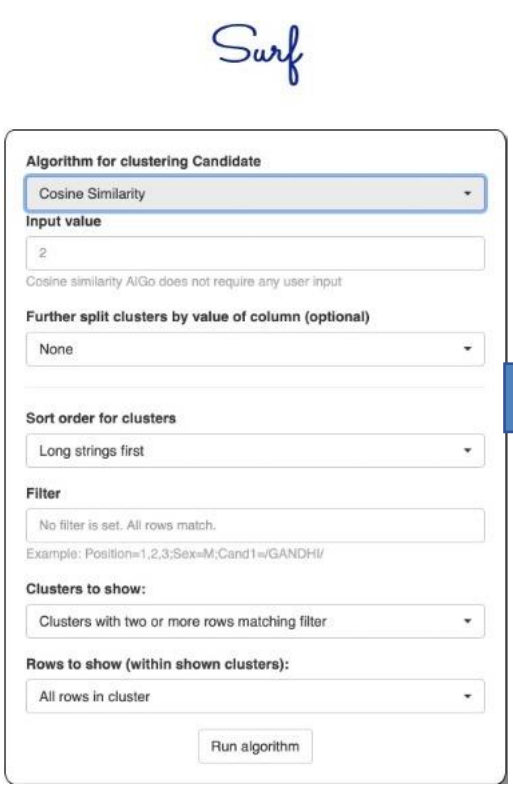


Upload dataset Window

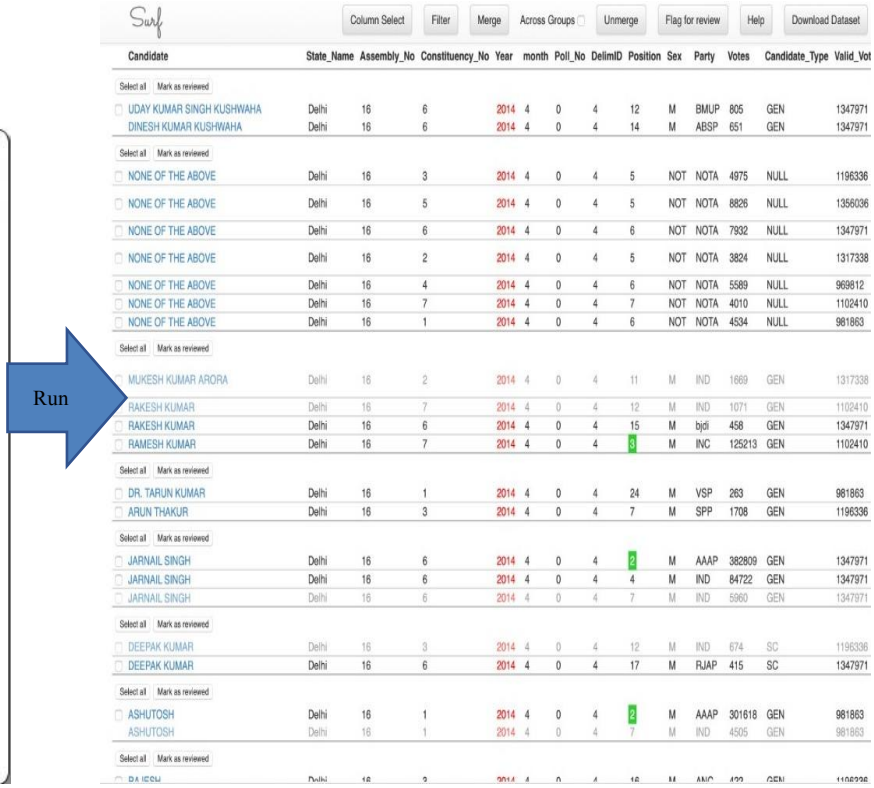


Filter Window (Can be empty as well)

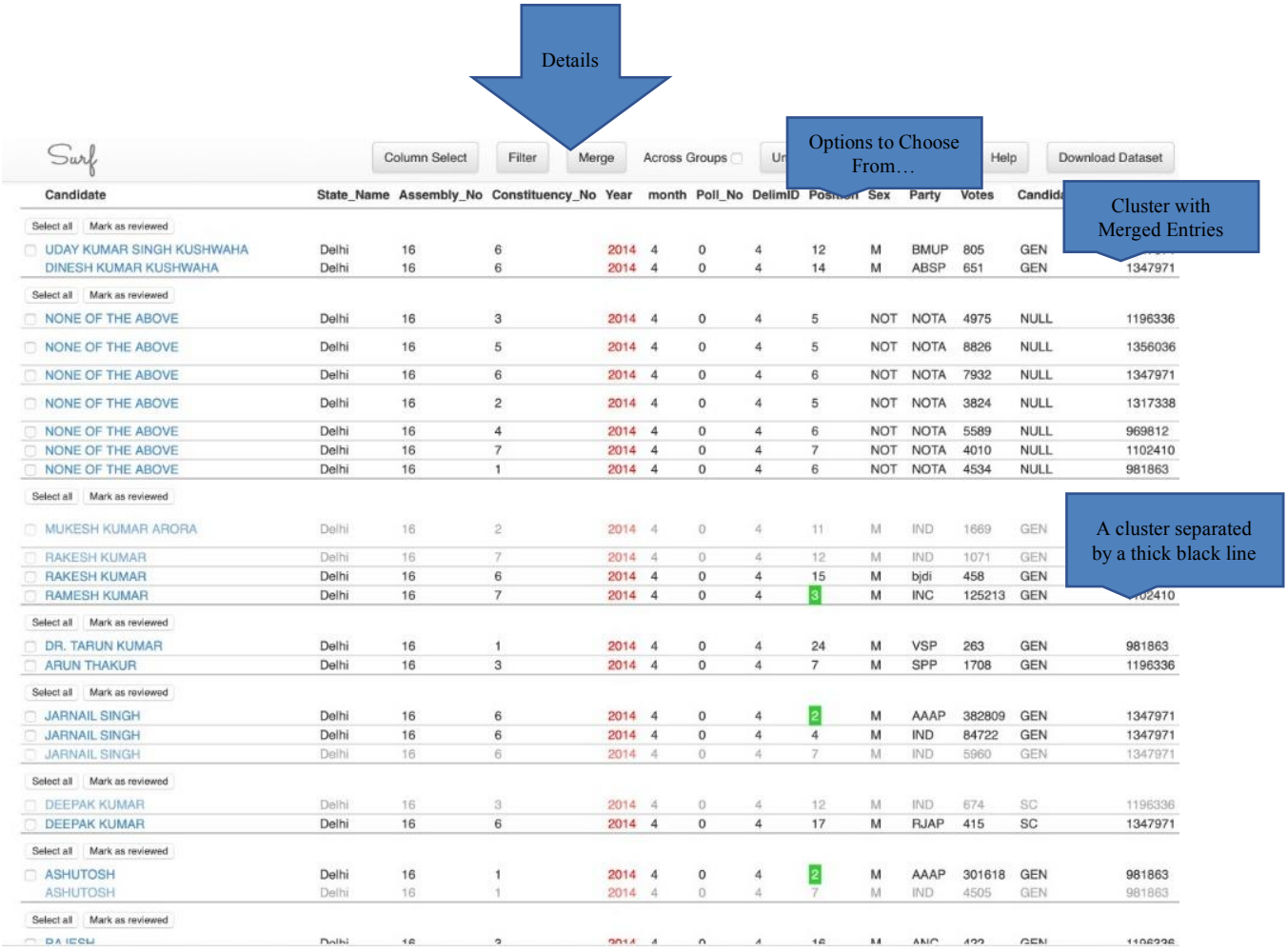
Select Algo



Choose Algorithm Window with its option



SURF Result window with Merge/Unmerge/Flag for Review/ Download Dataset Features at the top. Clusters as Shown.



SURF Window with all the details of the functionalities (Dataset: Delhi Assembly elections 2019 with InputVal = 95)

8. Understanding SURF codebase hierarchical table

The following charts of WebContent and MergeAlgorithms (src) explain the SURF codebase: -