

# SURF

Entity Matching and Resolution System for Indian Names by TCPD (Ashoka University)  
By Aditya Garg (IIITD)

## Table of Contents: -

<b>1. Introduction: -</b>	2
<b>2. Technical Background: -</b>	2
<b>3. My Work: -</b>	2
3.1 Cosine Similarity Clustering Algorithm	3
3.2 Edit Distance Clustering Algorithm	4
3.3 Compatible Names Clustering Algorithm	5
3.4 Review Algorithm	6
3.5 Space and Time Complexity Analysis	7
<b>4. Conclusion</b>	8
<b>5. What's Next</b>	8
<b>6. How to run SURF</b>	9
<b>7. How to run SURF user interface</b>	10
<b>8. Understanding SURF codebase hierarchical table</b>	12

## **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  $\cos\theta$  value in the computed cosine formula.

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

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: -

The screenshot displays the Surf application interface. On the left, a sidebar titled "Algorithm for clustering Candidate" shows the "Cosine Similarity" algorithm selected. The "Input value" is set to 2. Below this, there are options for "Further split clusters by value of column (optional)" (set to None), "Sort order for clusters" (set to Long strings first), a "Filter" section, "Clusters to show" (set to Clusters with two or more rows matching filter), and "Rows to show (within shown clusters)" (set to All rows in cluster). A "Run algorithm" button is at the bottom of the sidebar. A blue arrow points from the sidebar to the main data table on the right.

The main data table, titled "Surf", shows the results of the clustering algorithm for Delhi Assembly Elections-2019. The table has columns: Candidate, State\_Name, Assembly\_No, Constituency\_No, Year, month, Poll\_No, DelimID, Position, Sex, Party, Votes, Candidate\_Type, and Valid\_Vot. The table is divided into several sections, each with a "Select all" and "Mark as reviewed" link. The first section contains candidates UDAY KUMAR SINGH KUSHWAHA and DINESH KUMAR KUSHWAHA. The second section contains "NONE OF THE ABOVE" for various constituencies. The third section contains MUKESH KUMAR APORA, RAKESH KUMAR, RAKESH KUMAR, and RAMESH KUMAR. The fourth section contains DR. TARUN KUMAR and ARUN THAKUR. The fifth section contains JARNAIL SINGH, JARNAIL SINGH, and JARNAIL SINGH. The sixth section contains DEEPAK KUMAR and DEEPAK KUMAR. The seventh section contains ASHUTOSH and ASHUTOSH. The eighth section contains DAVESH.

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*  $\geq \text{Min}(\text{len}(\text{String 1}), \text{len}(\text{String 2}))$  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  $\leq$  “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: -

The image shows the Surf application interface. On the left is a sidebar with settings for the clustering algorithm. On the right is a table of election data for Bihar, 2015. A blue arrow points from the 'Long strings first' sort order setting to the table.

**Algorithm for clustering Candidate**

- Edit distance:
- Maximum edit distance:
- Edit distance 0 not included: ☐
- Further split clusters by value of column (optional):
- Sort order for clusters:
- Filter:
- Clusters to show:
- Rows to show (within shown clusters):
- 

**Table: Bihar Assembly Elections-2015**

Candidate	State_Name	Assembly_No	Constituency_No	Year	month	Poll_No	DelimID	Position	Sex	Party	Votes	Candidate_Type	Vali
<input type="checkbox"/> ANIRUDH PRASAD ALIAS SADHU YADAV	Bihar	17	19	2019	4	0	4	3	M	BSP	25039	GEN	972
<input type="checkbox"/> SURENDRA PRASAD YADAV	Bihar	17	36	2019	4	0	4	2	M	Rashtriya Janata Dal	333833	GEN	822
<input type="checkbox"/> ARAVIND KUMAR SHARMA	Bihar	17	19	2019	4	0	4	7	M	Bharatiya Jan Kranti Dal (Democratic)	8006	GEN	972
<input type="checkbox"/> ARVIND KUMAR SHARMA	Bihar	17	28	2019	4	0	4	9	M	Rashtriya Hind Sena	9291	GEN	103
<input type="checkbox"/> RAJESH KUMAR PASWAN	Bihar	17	38	2019	4	0	4	9	M	Aam Janta Party Rashtriya	11671	SC	957
<input type="checkbox"/> RAJNISH KUMAR PASWAN	Bihar	17	29	2019	4	0	4	30	M	IND	2602	GEN	103
<input type="checkbox"/> PRADEEP KUMAR SINGH	Bihar	17	15	2019	4	0	4	14	M	SHS	4321	GEN	105
<input type="checkbox"/> PRADEEP KUMAR SINGH	Bihar	17	9	2019	4	0	4	1	M	BJP	618434	GEN	116
<input type="checkbox"/> PRADIP KUMAR SINGH	Bihar	17	10	2019	4	0	4	15	M	SHS	3266	GEN	110
<input type="checkbox"/> RANDHIR KUMAR SINGH	Bihar	17	19	2019	4	0	4	2	M	Rashtriya Janata Dal	315580	GEN	972
<input type="checkbox"/> SANDIP KUMAR SINGH	Bihar	17	8	2019	4	0	4	8	M	IND	11534	GEN	111
<input type="checkbox"/> RAJ KUMAR SINGH	Bihar	17	7	2019	4	0	4	9	M	BSP	9066	GEN	106
<input type="checkbox"/> RAJIV KUMAR SINGH	Bihar	17	12	2019	4	0	4	14	M	IND	3403	GEN	115
<input type="checkbox"/> SAMIR KUMAR JHA	Bihar	17	11	2019	4	0	4	5	M	IND	8119	GEN	111
<input type="checkbox"/> SUJHIR KUMAR JHA	Bihar	17	15	2019	4	0	4	4	M	Yuva Krantikari Party	15843	GEN	105
<input type="checkbox"/> RAMCHANDRA SAH	Bihar	17	3	2019	4	0	4	10	M	Rashtriya Jansambhavana Party	6402	GEN	996

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)  $\geq 2$ , 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 =  $\text{MaxToken}(\text{String1}, \text{String2})$ , it indicates that the two strings are identical.

Surf

**Algorithm for clustering Candidate**

Compatible names

**Token overlap**

2

Group rows together if this number of tokens are common

**Ignore token frequency**

200

Frequency threshold in this dataset beyond which a token will be ignored

**Cluster substrings together**

☒

If checked, a name is clustered with a longer name of which it is a part.  
e.g., *RAM GOPAL* is clustered with *RAM GOPAL VARMA*  
The number of overlapping tokens does not matter.

**Allow initials**

☒

If checked, allows partial words or initials from one name to full words in the other.  
e.g., *M. GANDHI* is clustered with *MOHANDAS GANDHI*

**Further split clusters by value of column (optional)**

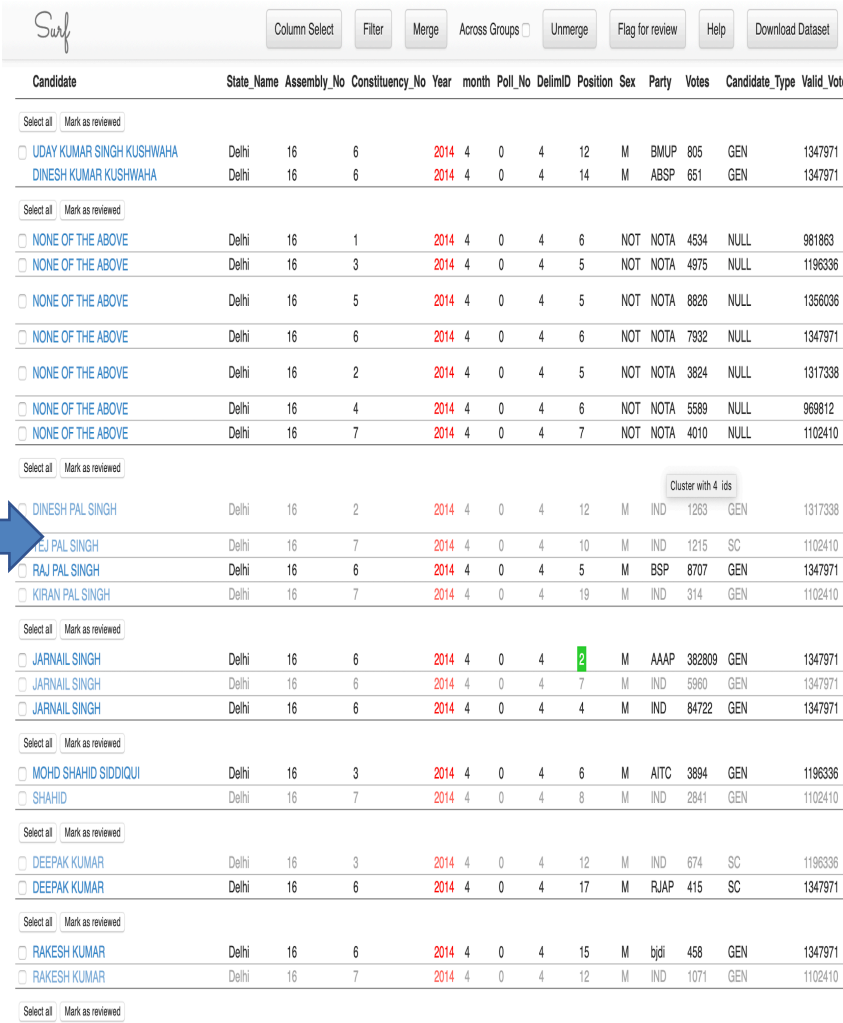
None

**Sort order for clusters**

Long strings first

**Filter**

No filter is set. All rows match.



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: -

The screenshot displays the Surf application interface. On the left, the 'Algorithm for clustering Candidate' is set to 'Review Algorithm'. Below this, there are options for 'Further split clusters by value of column (optional)' (set to 'None'), 'Sort order for clusters' (set to 'Long strings first'), a 'Filter' section (with a message 'No filter is set. All rows match.' and an example filter), 'Clusters to show:' (set to 'Clusters with two or more rows matching filter'), and 'Rows to show (within shown clusters):' (set to 'All rows in cluster'). A 'Run algorithm' button is at the bottom of this panel. A blue arrow points from the 'Run algorithm' button to the right panel.

The right panel shows the 'Review Algorithm Result' for the dataset of Delhi Assembly Elections-2019. It features a table with columns: Candidate, State\_Name, Assembly\_No, Constituency\_No, Year, month, Poll\_No, DelimID, Position, Sex, Party, Votes, Candidate\_Type, and Valid\_Votes. The table is divided into sections by 'Select all' and 'Mark as reviewed' buttons. The first section contains candidates like UDAY KUMAR SINGH KUSHWAHA and DINESH KUMAR KUSHWAHA. The second section contains DILDAR HUSSAIN BEG and DAN BAHADUR YADAV. The third section contains DINESH PAL SINGH and MOHD. ARIF SIDDIQUE. The fourth section contains MOHD SHAHID SIDDIQUI, BABU SINGH DUKHIYA, JAMESH BISHURI, RUBY YADAV, SANJAY KUMAR RAI, and NARESH KUMAR CHANDALIYA. The fifth section contains DR. HARSH VARDHAN and SHRICHAND TANWAR. The sixth section contains BIR SINGH SONI and INDER SINGH SANSI. The seventh section contains AJAY KUMAR KHEMKA, MUKESH KUMAR ARORA, DEEPAK KUMAR, AJAY MAKAN, PADMAJA KANDUKURI, RAKESH KUMAR, DEEPAK KUMAR, and RAKESH KUMAR. The table shows various details for each candidate, including their state, assembly number, constituency number, year, month, poll number, delimited ID, position, sex, party, votes, candidate type, and valid votes.

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  $\sim O(l^2n^2)$  time and  $\sim O(n^2)$  space to compute.
3. Review Algorithm: It takes  $\sim O(n)$  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  $\sim 20$ -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>

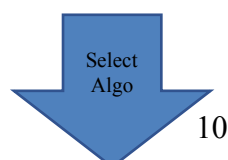
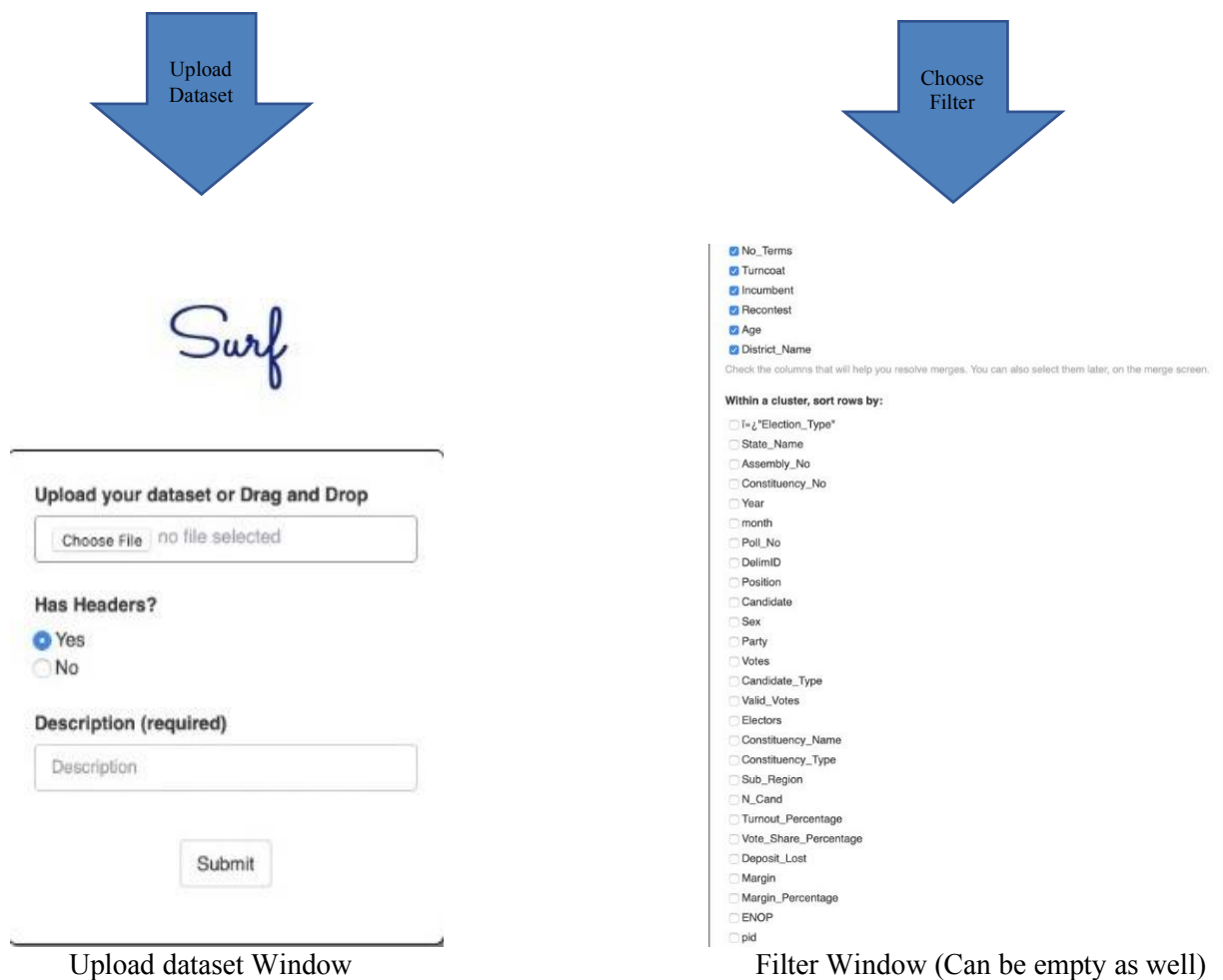
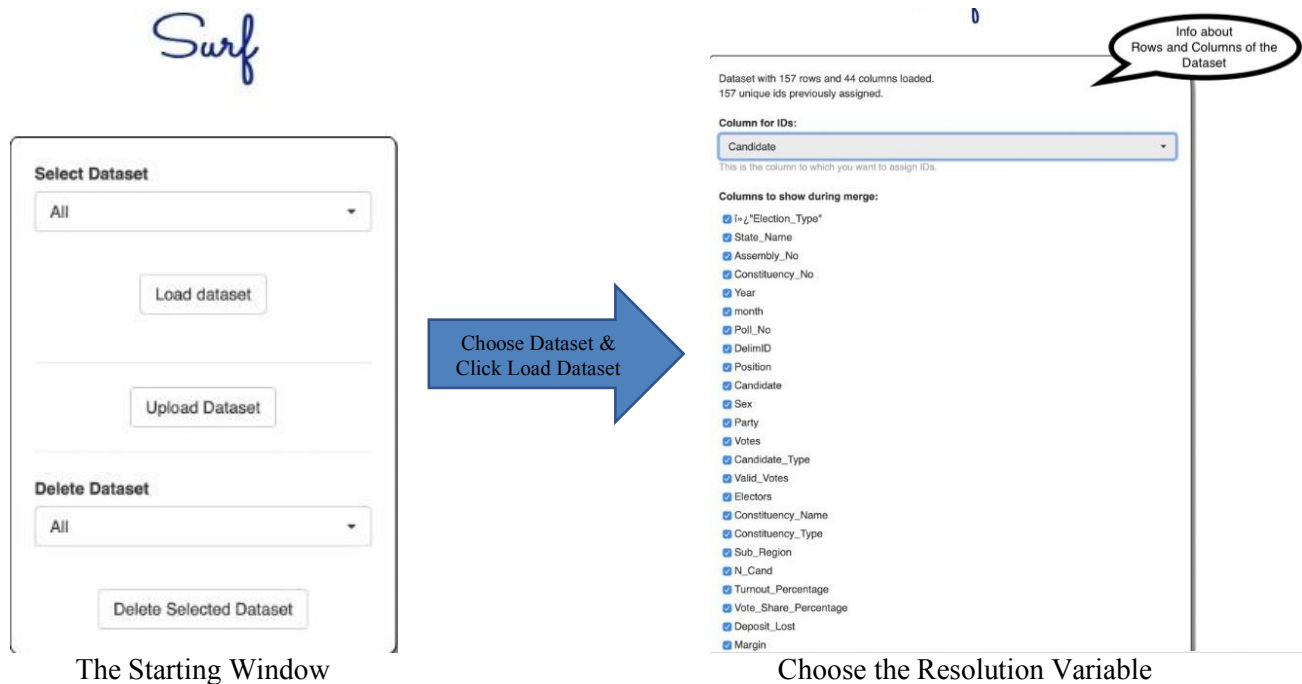
Apache Ant (ant) : <https://ant.apache.org/bindownload.cgi>

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: -



Surf

**Algorithm for clustering Candidate**

Cosine Similarity

Input value

2

Cosine similarity AI/Go does not require any user input

Further split clusters by value of column (optional)

None

Sort order for clusters

Long strings first

Filter

No filter is set. All rows match.

Example: Position=1,2,3;Sex=M;Cand1=GANDHI/

Clusters to show:

Clusters with two or more rows matching filter

Rows to show (within shown clusters):

All rows in cluster

Run algorithm

Choose Algorithm Window with its option



Run

Candidate	State_Name	Assembly_No	Constituency_No	Year	month	Poll_No	DelimID	Position	Sex	Party	Votes	Candidate_Type	Valid_Vot
<input type="checkbox"/> UDAY KUMAR SINGH KUSHWAHA	Delhi	16	6	2014	4	0	4	12	M	BMUP	805	GEN	1347971
<input type="checkbox"/> DINESH KUMAR KUSHWAHA	Delhi	16	6	2014	4	0	4	14	M	ABSP	651	GEN	1347971
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	3	2014	4	0	4	5	NOT	NOTA	4975	NULL	1196336
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	5	2014	4	0	4	5	NOT	NOTA	8826	NULL	1356036
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	6	2014	4	0	4	6	NOT	NOTA	7932	NULL	1347971
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	2	2014	4	0	4	5	NOT	NOTA	3824	NULL	1317338
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	4	2014	4	0	4	6	NOT	NOTA	5589	NULL	969812
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	7	2014	4	0	4	7	NOT	NOTA	4010	NULL	1102410
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	1	2014	4	0	4	6	NOT	NOTA	4534	NULL	981863
<input type="checkbox"/> MUKESH KUMAR ARORA	Delhi	16	2	2014	4	0	4	11	M	IND	1669	GEN	1317338
<input type="checkbox"/> RAKESH KUMAR	Delhi	16	7	2014	4	0	4	12	M	IND	1071	GEN	1102410
<input type="checkbox"/> RAKESH KUMAR	Delhi	16	6	2014	4	0	4	15	M	bjdi	458	GEN	1347971
<input type="checkbox"/> RAMESH KUMAR	Delhi	16	7	2014	4	0	4	3	M	INC	125213	GEN	1102410
<input type="checkbox"/> DR. TARUN KUMAR	Delhi	16	1	2014	4	0	4	24	M	VSP	263	GEN	981863
<input type="checkbox"/> ARUN THAKUR	Delhi	16	3	2014	4	0	4	7	M	SPP	1708	GEN	1196336
<input type="checkbox"/> JARNAIL SINGH	Delhi	16	6	2014	4	0	4	2	M	AAAP	382809	GEN	1347971
<input type="checkbox"/> JARNAIL SINGH	Delhi	16	6	2014	4	0	4	4	M	IND	84722	GEN	1347971
<input type="checkbox"/> JARNAIL SINGH	Delhi	16	6	2014	4	0	4	7	M	IND	5960	GEN	1347971
<input type="checkbox"/> DEEPAK KUMAR	Delhi	16	3	2014	4	0	4	12	M	IND	674	SC	1196336
<input type="checkbox"/> DEEPAK KUMAR	Delhi	16	6	2014	4	0	4	17	M	RJAP	415	SC	1347971
<input type="checkbox"/> ASHUTOSH	Delhi	16	1	2014	4	0	4	2	M	AAAP	301618	GEN	981863
<input type="checkbox"/> ASHUTOSH	Delhi	16	1	2014	4	0	4	7	M	IND	4505	GEN	981863
<input type="checkbox"/> DA IESU	Delhi	16	3	2014	4	0	4	16	M	ANC	423	GEN	1102410

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



Details

Candidate	State_Name	Assembly_No	Constituency_No	Year	month	Poll_No	DelimID	Position	Sex	Party	Votes	Candidate_Type	Valid_Vot
<input type="checkbox"/> UDAY KUMAR SINGH KUSHWAHA	Delhi	16	6	2014	4	0	4	12	M	BMUP	805	GEN	1347971
<input type="checkbox"/> DINESH KUMAR KUSHWAHA	Delhi	16	6	2014	4	0	4	14	M	ABSP	651	GEN	1347971
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	3	2014	4	0	4	5	NOT	NOTA	4975	NULL	1196336
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	5	2014	4	0	4	5	NOT	NOTA	8826	NULL	1356036
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	6	2014	4	0	4	6	NOT	NOTA	7932	NULL	1347971
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	2	2014	4	0	4	5	NOT	NOTA	3824	NULL	1317338
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	4	2014	4	0	4	6	NOT	NOTA	5589	NULL	969812
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	7	2014	4	0	4	7	NOT	NOTA	4010	NULL	1102410
<input type="checkbox"/> NONE OF THE ABOVE	Delhi	16	1	2014	4	0	4	6	NOT	NOTA	4534	NULL	981863
<input type="checkbox"/> MUKESH KUMAR ARORA	Delhi	16	2	2014	4	0	4	11	M	IND	1669	GEN	1317338
<input type="checkbox"/> RAKESH KUMAR	Delhi	16	7	2014	4	0	4	12	M	IND	1071	GEN	1102410
<input type="checkbox"/> RAKESH KUMAR	Delhi	16	6	2014	4	0	4	15	M	bjdi	458	GEN	1347971
<input type="checkbox"/> RAMESH KUMAR	Delhi	16	7	2014	4	0	4	3	M	INC	125213	GEN	1102410
<input type="checkbox"/> DR. TARUN KUMAR	Delhi	16	1	2014	4	0	4	24	M	VSP	263	GEN	981863
<input type="checkbox"/> ARUN THAKUR	Delhi	16	3	2014	4	0	4	7	M	SPP	1708	GEN	1196336
<input type="checkbox"/> JARNAIL SINGH	Delhi	16	6	2014	4	0	4	2	M	AAAP	382809	GEN	1347971
<input type="checkbox"/> JARNAIL SINGH	Delhi	16	6	2014	4	0	4	4	M	IND	84722	GEN	1347971
<input type="checkbox"/> JARNAIL SINGH	Delhi	16	6	2014	4	0	4	7	M	IND	5960	GEN	1347971
<input type="checkbox"/> DEEPAK KUMAR	Delhi	16	3	2014	4	0	4	12	M	IND	674	SC	1196336
<input type="checkbox"/> DEEPAK KUMAR	Delhi	16	6	2014	4	0	4	17	M	RJAP	415	SC	1347971
<input type="checkbox"/> ASHUTOSH	Delhi	16	1	2014	4	0	4	2	M	AAAP	301618	GEN	981863
<input type="checkbox"/> ASHUTOSH	Delhi	16	1	2014	4	0	4	7	M	IND	4505	GEN	981863
<input type="checkbox"/> DA IESU	Delhi	16	3	2014	4	0	4	16	M	ANC	423	GEN	1102410

Options to Choose From...

Cluster with Merged Entries

A cluster separated by a thick black line

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: -

