Spring 2013

Enhanced Clustering of Technology Tweets

Ananth Gopal
San Jose State University

Enhanced Clustering of Technology Tweets

A Writing Project Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Ananth Gopal

2013

ABSTRACT

Enhanced Clustering of Technology Tweets

Ever since Twitter has been widely accepted and has become an immensely popular micro blogging website, it is being used as a primary source of news; be it related to sports, entertainment, politics or technology by several users. It has been proven earlier that the elimination of stop words has a positive impact on the clustering of technology related tweets. The focus of this paper is to enhance the quality of clustering of the technology related Tweets by developing a semi-automated approach to eliminating stop words and by making use of a combination of Canopy and K-means clustering algorithms. The paper also details an algorithmic approach to determine the threshold values for Canopy clustering.

ACKNOWLEDGEMENTS

I would like to sincerely thank Dr. Teng Moh, my project advisor for his guidance and support.

Table of Contents

LIST OF FIGURES

Chapter 1

Introduction

Twitter has been widely accepted and has become an immensely popular micro blogging website. The content shared on this website has to be precise and concise in order to avoid exceeding the limit of 140 characters per Tweet [1]; yet convey whatever is intended correctly. This limitation is a great advantage for users who want to stay updated with the latest trends in technology or read about the current happening in sports, entertainment or politics in brief. Owing to these facts, Twitter has now become one of the main sources of news and users regularly perform searches on Twitter for the same.

Clustering the Tweets and then building an inverted index from the clustered collection would result in returning better search results to the user. Whenever a search is performed, there are thousands of results returned that are sorted by how similar they are to the search query. Clustering fundamentally groups the pages based on the content of each page, resulting in each group consisting of pages similar to each other. Each particular cluster could capture a specific aspect of the query.

Another advantage of performing searches on clustered data is that it would make each search more exploratory, i.e. the results returned for each search could be further broken down into subcategories [6]. For instance consider the case of a search query “movie”. There are several subcategories to the results returned for this search query. The movie reviews, the nearest theaters, different genres of movies etc. are some of the subcategories for the search. The users would now have an option to go more specific into the search that interests them.

Stop words are function words that do not contribute much to the clustering but are present abundantly in the text as they are required to structure the English language. For instance terms such as ‘the’, ‘is’, ‘at’, ‘which’, ’on’ etc. are stop words. However, these terms are present in almost all the documents as Tweets would completely lose their meaning if none of these terms were included. As a result of being present in numerous Tweets, when clustered, the stop words regularly appear as the top terms of several clusters.

Eliminating stop words from the text corpus before clustering significantly improves the quality of clustering of technology Tweets [1]. Identifying the stop words is quite a challenge as it is not possible to eliminate all the stop words. The approach followed by Surya Bhagavat and Teng Moh, has been completely manual. The author selects a set of stop words from a list of top 5 terms in each cluster. Selected terms from these top terms are then eliminated from the next round of clustering.

One of the objectives of this paper is to automate the stop words identification process. However, during the stop words elimination process, several terms shall be treated as custom stop words which might not actually be “stop words” either by definition or from the stand point of the English language. For instance, terms such as “Awesome”, “Great”, “Thanks” etc. are used very commonly and abundantly, but, they too do not contribute much to clusters relating to technology. These terms shall hence be treated as custom stop words. This makes human intervention necessary and hence the process requires users to select specific terms which will be treated as “custom stop words” and eliminated from the next round of clustering.

The clustering algorithm used by Surya Bhagavat and Teng Moh, is the K-means clustering algorithm. Though K-means clustering algorithm when used with Cosine distance measure works well for text data [2], the initial seed for the clustering is selected randomly by the algorithm. This can lead to inconsistencies in the clustering [2]. In order to resolve this issue, Canopy clustering can be executed on the data and its output can be fed to K-means as an input for clustering [2].

Though Canopy clustering yields better quality of clustering when followed by the use of K-means clustering algorithm, canopy clustering has its limitations. Canopy clustering requires that two threshold values t1 and t2 be provided as input which is generally determined by trial and error. The paper discusses an algorithmic approach developed to determine the threshold values for Canopy clustering.

Chapter 2

Related Work

Mining of Twitter data has been performed with several purposes. For instance improving the search results returned to the user, identifying primary interests in users, identifying products and services that users are fond off etc. are some uses for mining Tweets. Identifying the services used by the twitter users aids the third party services to alter their own services to be more streamlined towards the users' requirements. Considering these, mining of twitter data has been a topic of research for a significant period of time.

Some of the interesting approaches taken include the approach taken by Qing Chen et al., Swit Phuvipadawat et al. and *Clustering of Technology Tweets and the Impact of Stop Words on Clusters, ACMSE'12, March 29–31, 2012 by Surya Bhagavat and Teng Moh.*

In the paper by *Surya Bhagavat and Teng Moh*, a study of the impact of eliminating stop words on the clustering of technology related Tweets has been discussed. The paper compares results of the quality of clusters after eliminating full set of English stop words, eliminating only the complete set of MySQL stop words and finally quality of cluster after eliminating a specific set of custom stop words.

The observations made and results recorded by *Surya Bhagavat and Teng Mohs'* paper serve as a starting point for this paper. The data was collected by Surya Bhagavat was with the use of Twitter4j, making use of Amazon Rackspace to set up multiple virtual machines to work around the limitations of Twitter4j on the number of API calls one machine can make in a single

day. Also the stop words identification process has been manual with the top terms in each cluster being collected and manually scanned to eliminate custom stop words. TF-IDF weights have been used to evaluate and compare the quality of clustering in all cases including eliminating entire set of English stop words, entire set of MySQL stop words and finally only eliminating a custom set of stop words.

The observations record that the use of complete set of English or MySQL stop words results in meaningless clusters proving the importance of selecting appropriate stop words to eliminate before clustering [1].

Chapter 3

Problem Formulation

This section presents the problems identified and resolved by this paper.

1) Manual stop words identification

The stop words that need to be eliminated are being currently identified manually by scanning through the top five terms from each cluster. This process is cumbersome and is highly time consuming. An approach should be designed to at least partially select stop words algorithmically.

2) Stop words to be eliminated vary by collection

The present approach taken has proven that eliminating stop words has a positive impact on quality of clusters. However, if the stop words identified for elimination for a specific collection are also eliminated on a different collection, it might not be as effective as expected. This implies that a new set of stop words have to be identified for different collections and the current process.

3) K-means algorithm might not produce optimal number of clusters

The K-means algorithm works well for text data clustering together with Cosine distance measure [2] but it is not guaranteed to have optimal centroids because the algorithm chooses random initial centroids to cluster the collection.

4) Data gathering

The data collection process involves set up of multiple virtual machines to circumvent the restrictions on the number of API calls from one machine enforced by Twitter4j. This process needs to be simplified.

Chapter 4

Solution

This chapter briefly presents the solutions to every point in the problem formulation specified above.

4.1 Tool for Stop Word selection – Reduces manual intervention

The manual approach taken to select stop words for a collection can be at least semi-automated with the use of a tool developed to identify stop words intelligently. The developed tool significantly reduces the human interaction in the process of identification of stop words from the collection. The tool renders the process less cumbersome and greatly reduces the time consumed in identifying appropriate.

There are primarily two main components to the tool based stop word identification process:-

Automatically eliminate MySQL Stop Words

Top five terms from each cluster are collected. The complete set of MySQL stop words cannot be eliminated from the collection as it will result in meaningless clusters [1]. Only those terms present in the top terms from each cluster which are also present in the MySQL stop words list are selected to be eliminated as stop words to the corpus. The MySQL stop words are eliminated on every round of clustering as long as the “Eliminate MySQL Stop Words” radio is checked.

The algorithm used to apply MySQL stop words is described below:

```
Eliminate MySQL stop words:
function eliminateMySQLStopWords(): stopWordList
if(isMySQLRadio)
    SET stopWordsList to terms in termsandweights & MySQLStopWords;
    return (stopWordsList)
else
    Return (emptyList)
endif
endFuncion
```

Eliminate Custom Stop Words selected by user

Apart from the MySQL stop words that are conditionally eliminated by the tool, there are a set of terms that are presented to the user from which the user can identify terms which will also be treated as stop words. These terms that are presented to the user are terms that are not a part of the MySQL stop words as they will be eliminated automatically by the tool. All the terms selected by the user will be appended to the list of stop words and eliminated from the collection in the next round of clustering. Some terms which are directly related to technology are terms that are used freely and regularly in

communication, but might not really contribute to the clusters; these terms need to be chosen as stop words.

Selecting Custom Stop Words

The Custom Stop Words that the user identifies should satisfy a few conditions to qualify as a Stop Word. These Stop Words however entirely depend on what the user decides on and hence might not be the best stop words selected. The tool however shall maintain the best clustering achieved so far. The clustering of tweets and stop word identification is based on the feedback from the tool to the user in real time in terms of the scaled average inter cluster distance. This feedback mechanism and the maintenance of best achieved clustering ensure that the quality of clustering is better than what it was without eliminating any stop words. The following are some guidelines as to how custom Stop Words are identified:

- Think from the Point-Of-View of a search, i.e. do not select terms that are more likely to appear as a part of a search term
- In several Tweets, it is possible that there might be a reference to a user name, select such terms
- Some Tweets contain swear words or unparliamentarily words that occur frequently and might appear in top terms of a cluster, select such terms too
- Also, some terms like “Users”, “User”, “amazing”, “awesome”, “great” etc. are also used extensively but do not contribute to clustering, such terms should be eliminated

- Select terms that are abbreviations of terms that are not related to technology, for instance terms like “lol”, “rofl” etc.

Hence the overall stop words identification process is a combination of both MySQL stop words applied directly by the tool and the custom stop words identified by the user. The complete stop word elimination process is presented below:

```
Eliminate stop words:
function eliminateStopWords(): stopWordsList
|displayTerms| <- |topTerms| \ |mySQLStopWords|
|customStopWordsList| <- terms identified by user
if(isMySQLRadio)
  |mySQLStopWordsList| <- eliminateMySQLStopWords()
  |stopWordsList| <- |mySQLStopWordsList| U
|customStopWordsList|
  return |stopWordsList|
else
  return |stopWordList|
EndIf
EndFunction
EndFunction
```

4.2 Stop Words by collection

The tool based approach requires that the user choose the location at which the data set to be clustered is present. First round of clustering is performed with no stop words

eliminated and a set of terms from the top terms in each cluster which are not present in MySQL stop words list is displayed to the user. In subsequent rounds of clustering the user selects custom stop words to be eliminated from the collection and the tool eliminates MySQL stop words automatically if permitted by the user. The stop words that have been identified will depend on the data set chosen and hence the stop words that need to be eliminated are identified for the specific collection. The tool also maintains the dictionary post elimination of stop words for the best quality of clustering achieved, thereby ensuring that too many terms are not eliminated rendering the clusters meaningless.

4.3 K-means centroids might not be optimal

K-means algorithm is quite efficient and performs well with text clustering when combined with the Cosine distance measure [2]. However the quality of clustering relies heavily on the initial centroids that the K-means algorithm chooses. The K-means algorithm's implementation requires that the number of clusters be mentioned and passed as an input parameter (k). It is not possible to determine precisely the number of clusters that is needed to be generated for the corpus. This affects the resultant quality of the clusters generated by the K-means algorithm. The solution to this is to make use of Canopy clustering to determine initial centroids before executing K-means clustering by passing output of canopy clustering as an input to the K-means algorithm. It is proven that the estimates by the Canopy clustering algorithm significantly improves the quality of clustering of the clusters generated by K-means algorithm [2].

Canopy clustering would require that initial threshold values be provided which is determined by trial and error. In order optimize the time taken to identify thresholds an

algorithmic approach has been designed to determine threshold. Observations made by executing Canopy clustering on different data sets have been used to identify an initial value for threshold. Subsequently a number of executions of Canopy algorithm yields acceptable threshold values.

4.4 Data Collection

The process for gathering Tweets that has been mentioned the paper by *Surya Bhagavat and Teng Moh*, is cumbersome and time consuming. The manner in which data has been collected requires multiple virtual machines be set up to circumvent the restriction of Twitter4j API on the number of API calls each machine can make per day.

Alternately, a solution to this is to make use of Otter Java API from Topsy [7]. Topsy is one of many websites which maintain Tweets for over a period of time by streaming Tweets on a regular basis and persisting them in a database. Twitter itself does not maintain Tweets over a very long period of time as it has restrictions on the number of tweets to be maintained per timeline.

Otter Java API allows Tweets from the past two years to be collected. The restriction on the number of API calls is tied to an API Key. Registered users can make up to 7000 calls per day for free and in each call can collect up to 100 tweets. This is a convenient method for collecting Tweets.

Chapter 5

Design and Implementation

We have implemented a Tool that partially automates the elimination of stop words and also retains the best quality of clustering from multiple executions of the clustering algorithm during the process based on the best scaled inter cluster distance on each run. This section presents in detail the design and implementation of the tool, detailed data gathering procedure and the multiple algorithms developed for the tool. Also, a new data collection technique has been discussed in full detail in this chapter.

In order to eliminate the various issues discussed earlier, we have built a Tool that utilizes Canopy clustering to finalize the clustering procedure. There are two main processes in this tool:-

1) Stop words identification and elimination process
2) Finalize clustering process

The following sections present a detailed description of the overall procedure, the stop elimination process and the clustering finalization process.

Tool Based Clustering Procedure

The tool has been designed to ensure the best clustering possible for a collection is achieved at the end of the procedure. The clustering procedure starts with the user selecting a dataset, followed by a stop words identification and elimination process and concludes with the

finalization of clustering. During the finalization phase, thresholds required by canopy algorithm are determined with no input from the user. Below is a flow chart of the overall process:-

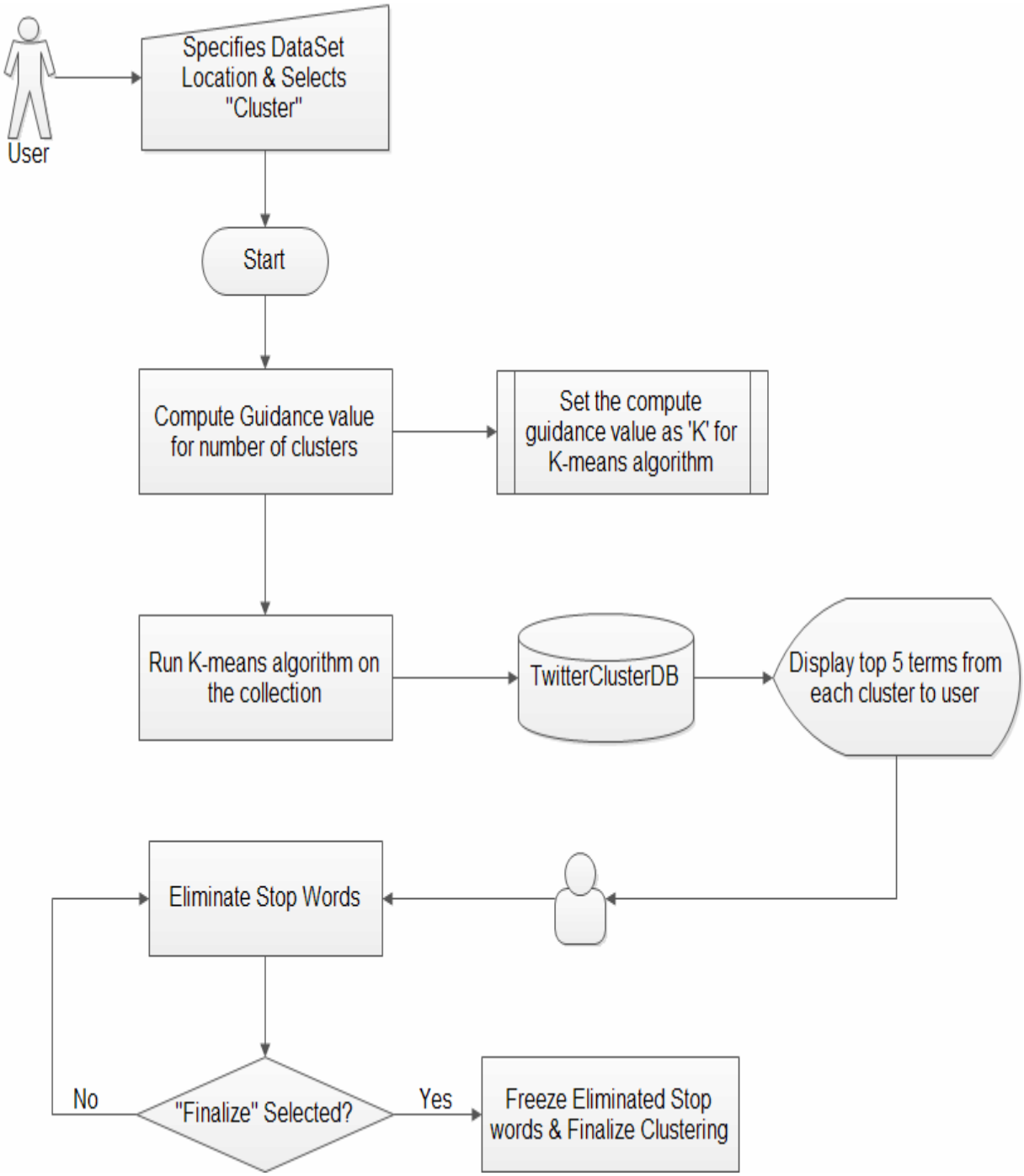

Figure 1: Overview of clustering procedure

The procedure for clustering starts with the user specifying the dataset. The user can point the tool to a location on the system where the data is stored. Once the "Cluster" button is selected, the tool performs the following functions:

Compute Guidance Value for Number of Clusters

The K-means algorithm requires that the value of "K", which is the number of clusters to be formed for the collection to be provided as an input parameter. Alternately, the output of a different clustering algorithm can be provided as the input to K-means. However, since the stop words to be eliminated are determined incrementally, making use of multiple algorithms on each run would make it a highly time consuming process. For this reason, the first approach of providing a value for number of clusters has been used in this paper.

The number of clusters to be formed for a collection cannot be determined accurately. To do so, it would require us to know the nature of the data present. It is possible to approximate the number of clusters based on the size of the data set. A rule of the thumb can be used to determine the number of clusters to be formed from the collection.

As per the rule of thumb, the equation for calculating the number of clusters is

$$k \cong \sqrt{\frac{n}{2}}$$

Where k is number of clusters and n is the number of data points present in the collection to be clustered [17].

In this case the number of data points are the number of files present as each file contains exactly one Tweet. This equation is present has been presented in the book Kanti Mardia et al. (1979). Multivariate analysis, Academic press [17]. This is an extremely simple method to estimate the number of clusters required to be generated by the tool.

The tool requires that the user examine the top five terms from each cluster and identify "custom stop words" from the collection. These terms apart from the MySQL stop words present in the top five terms collected will be eliminated from the collection before the next round of clustering. Depending upon the size of the dataset, the number of terms that the user has to examine and choose from is directly proportional to the number of clusters.

Using the formulae, even for a collection of 20000 documents there are only 100 clusters generated from it. This results in the user having to examine a maximum of just 500 terms to eliminated custom stop words.

Execute K-Means Clustering Algorithm

The paper uses a third party tool, Apache Mahout, a machine learning library, to execute the clustering algorithm. In order to execute the K-means algorithm on a text collection, Mahout requires the text documents to be in the form of vectors. For this purpose, it is necessary to convert the text documents and prepare them for clustering.

To convert the text documents into vectors, we use TF-IDF scores and replace the term with the score for each term in every document. TF-IDF scores are a numerical statistic which

indicates how important a term is to a document in a corpus []. TF indicates importance of term in specific document while IDF represents the importance of a term relative to entire corpus [].

Converting text to vectors is a two-step process:

➔ *Convert text to Sequence Files*

The machine learning library provides a utility to perform this operation. The following command converts the Text documents into Sequence Files [2]

```
$Mahout_HOME/bin/mahout seqdirectory -c UTF-8
-i <<Location of Input Text Documents>>
-o <<Location where the Sequence File should be generated>>
```

➔ *Covert Sequence Files to TF-IDF Vectors*

The sequence files thus generated should now be converted into TF-IDF vectors which can be fed as an input to the clustering algorithm. The following command is used to perform this operation [2]

```
java
org.apache.mahout.vectorizer.SparseVectorsFromSequenceFiles
--input << Location of the Sequence file generated >>
--output << Location to generate TF-IDF vectors>>
--analyzerName << Class containing the stop words >>
--namedVector
--minDF 2 --maxDFPercent 99
```

```
        --weight TFIDF  --chunk 100
```

From the arguments that are passed, the analyzerName is should contain a set of stop words that need to eliminated from the collection. These terms will not feature in the TF-IDF vectors that will be generated. Since we determine stop words incrementally based on user input, it is necessary to convert the SequenceFile into a vectors each time clustering algorithm is executed.

Once the TF-IDF vectors have been generated, the clustering algorithm can be executed on the vector file. Invoking the following command executes the K means algorithm.

```
java org.apache.mahout.clustering.kmeans.KMeansDriver

-i << Location where TF-IDF vectors have been generated >>
-c << Location of initial clusters generated >>
-o << Location where output is to be produced >>
-cd 0.5

-x 25

-k << Guidance Value Computed >>

-cl
-dm << Distance Measure to be used >>
```

On successfully completing the clustering, Mahout produces a cluster output file. Mahout provides a utility to dump the cluster output to a human readable format known as the Cluster Dumper. The generated cluster output file contains the top terms in each cluster, followed by the

actual contents of the each cluster. The number of top terms to be collected can be specified as a part of the arguments passed to the utility.

The cluster dumper can be invoked with the following command

```
java org.apache.mahout.utils.clustering.ClusterDumper
-s << Location of the clusters file generated >>
-p << Location of the cluster points file generated >>
-n << Number of top terms collected from each cluster >>
-d << Location of dictionary file >>
-dt sequencefile
-o << Location where cluster output is to be generated>>
```

The following figure shows the output produced by the cluster dumper utility.

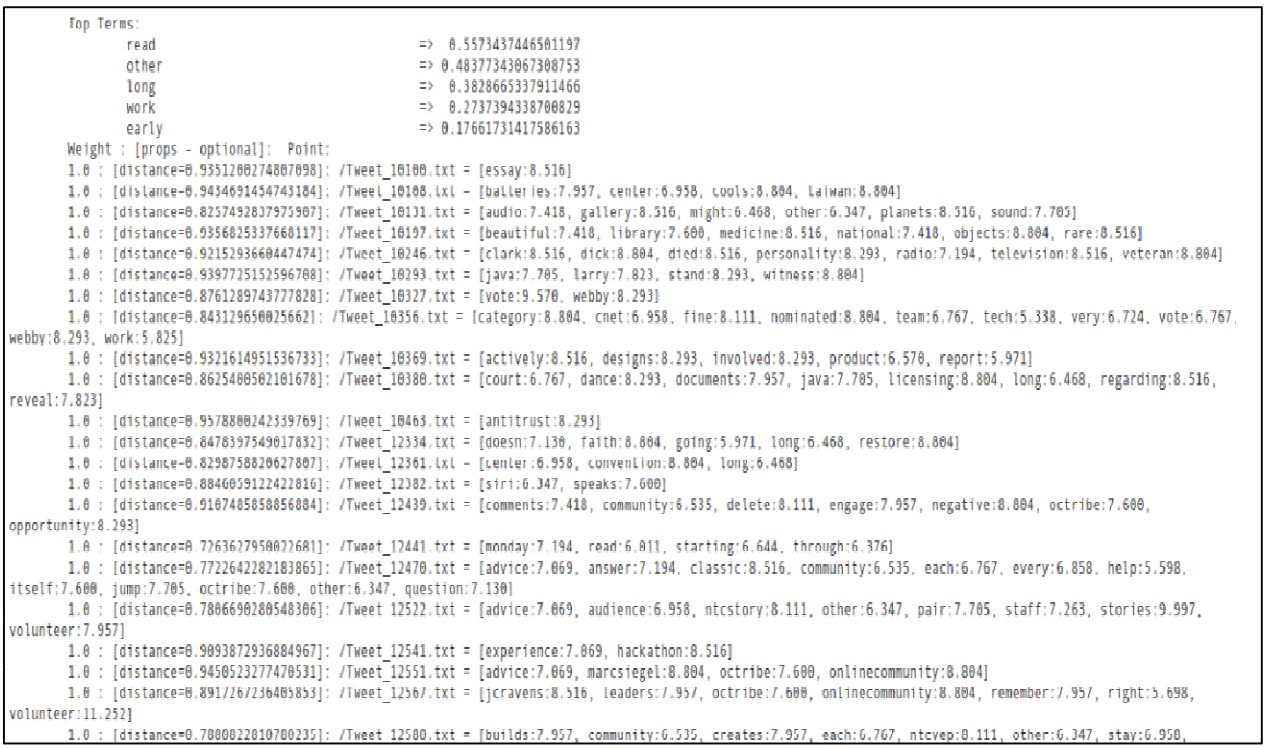

Figure 2: Cluster Dumper Output

The top terms are generated with their corresponding TF-IDF score. Each line following the top terms contains one Tweet. Every term in each Tweet is followed by the TF-IDF score and presented as above.

We have developed a cluster dumper reader which collects the top terms and persists them into the database. The cluster dumper reader also splits the cluster dumper output file and generates a separate file for each cluster.

A MySQL database is used to persist the top terms collected from each cluster. A table by name TermsAndWeights is used to collect the terms. The following is the description of the table.

Field	Type	Null	Key	Default	Extra
Term	varchar(100)	NO	PRI	NULL	
Weight	double	NO		0	
Field_1	varchar(100)	YES		NULL	

Figure 3: TermsAndWeights Table Description

Term – This field will contain all the top terms collected from each cluster

Weight – Contains the corresponding TF-IDF weight for the Term

Field_1 – An additional field to note the cluster in which the term is present

The tool collects the top five terms from each cluster. As mentioned before now the user has to examine these terms and identify custom stop words.

The tool provides a simple user interface that permits users to make use of check boxes to select terms that will need to be eliminated from the next round of clustering. A screen shot of the tool displaying the top terms to the user is shown below.

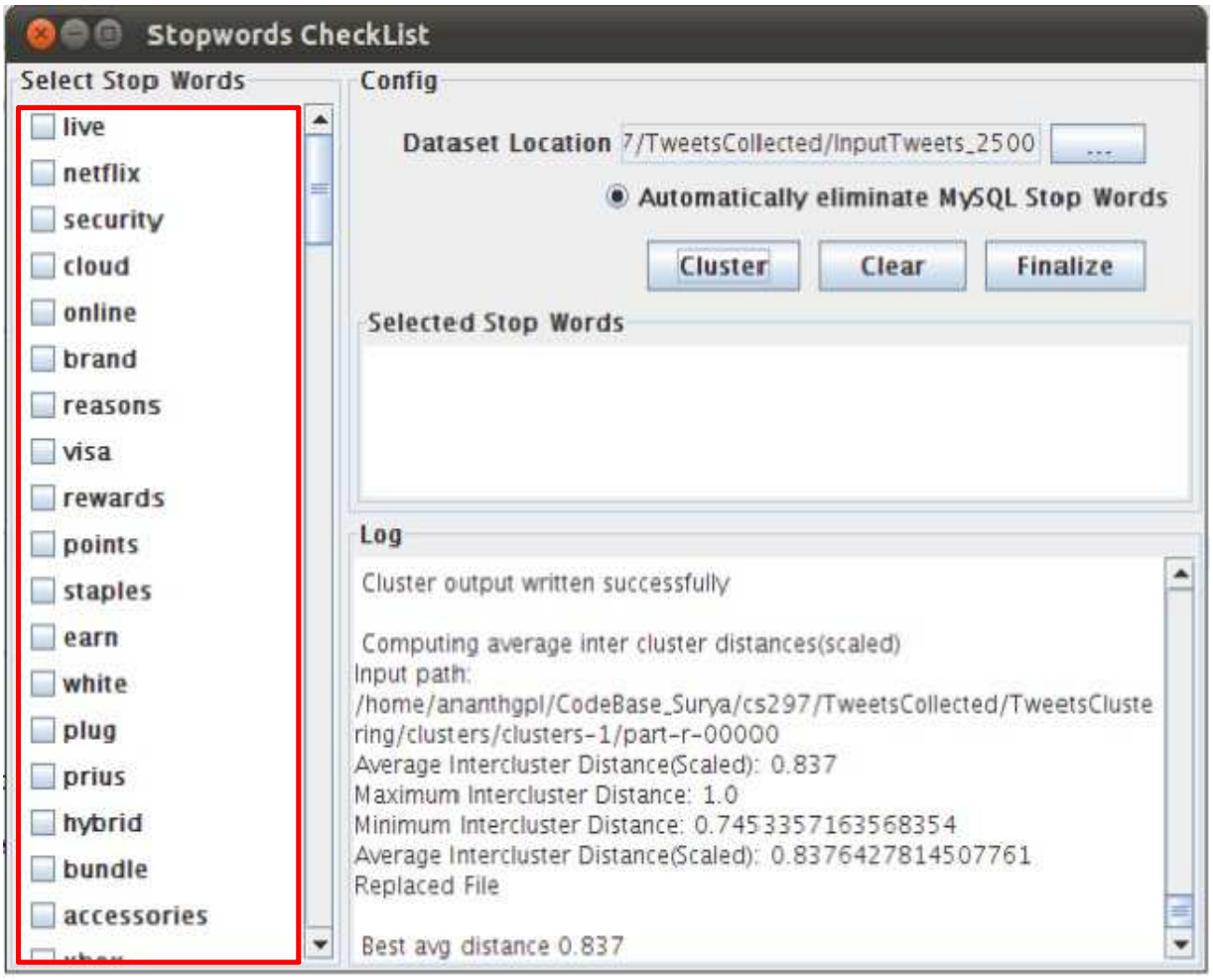

Figure 4: Top terms in each cluster

The highlighted portion on the left corner shows the top terms collected and displayed by the tool. The selected stop words will be displayed separately for the user to review. Unchecking the radio for a specific term will cause the unchecked term to not be eliminated from the next round of clustering.

Eliminate stop words

At this point the tool has executed the clustering algorithm with the computed value of k and displays the top terms in each cluster to the user. In the first round of clustering the "Automatically eliminate MySQL Stop Words" radio does not affect the clustering. This radio is checked by default which causes the tool to intelligently eliminate specific MySQL stop words from the corpus; provided the user retains the radio as checked. The overview of the stop words elimination process is show in the flow chart below:

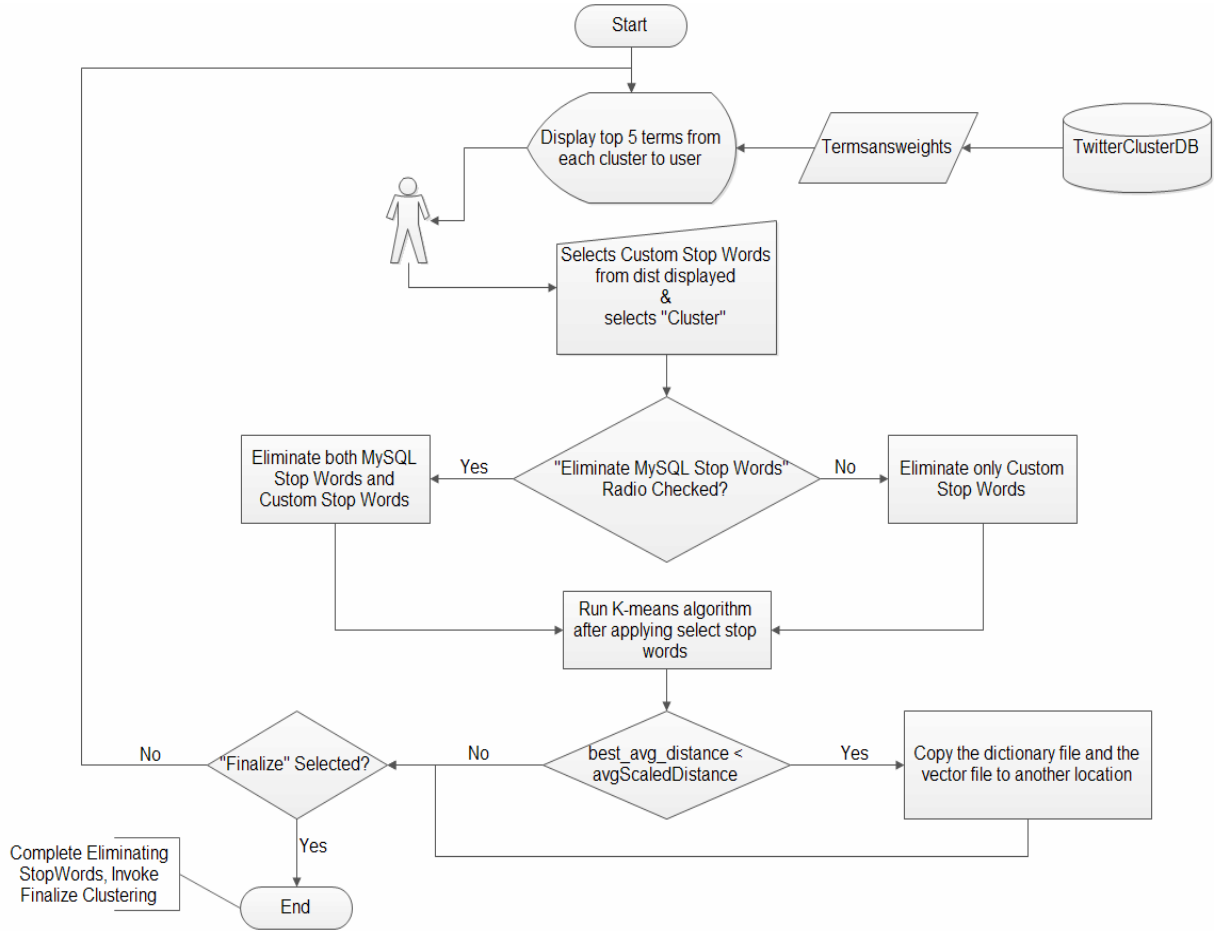

Figure 5: Procedure to Eliminate stop words

The stop word elimination process starts with the user pointing the tool to the location of the dataset. The tool executes the K means algorithm with a guidance value computed and set to K. There are two parts to the process:

- Automatically eliminating MySQL stop words
- Eliminate custom stop words

Automatically eliminate MySQL stop words

The process to eliminate MySQL stop words does not involve any human interaction. The tool makes use of two database tables to select the stop words and eliminates them when the radio button is checked. The two tables are the TermsAndWeghts table and the MysqlStopWords table. The former table has been described in the earlier section and the details of the latter are below:

Field	Type	Null	Key	Default	Extra
SlNo	int(11)	NO	PRI	NULL	auto_increment
StopWord	varchar(50)	YES		NULL	

Figure 6: MySQLStopWords Table Description

SlNo – Numerical value, unique for every

StopWord – Contains complete set of MySQL stop words

The tool performs a join on the TermsAndWeights table and the MySQLStopWords table based on the terms and eliminates them from the collection. The entire set of MySQL stop words being eliminated from the collection will adversely affect the clustering quality [1] and hence it cannot be done. This methodology identifies only the MySQL stop words which appear in the top terms of clusters thereby eliminating only those terms which contribute to clusters but are not actually related to technology.

Eliminate Custom Stop Words

Apart from the MySQL stop words eliminated automatically, the user also selects a set of terms from those displayed to be eliminated from the next round of clustering. These terms will persisted in the database in the CustomStopWords table. This table contains exactly one field which contains all the terms identified by the user. The final set of stop words to be eliminated will be the terms present in both the CustomStopWords table and the terms chosen by the tool as MySQLStopWords from the collection.

Terms present in the CustomStopWords are combined with the terms chosen by the tool and are persisted into another table, the StopWordsApplied table. This is necessary as these terms cannot be eliminated till the next round of clustering. These terms need to be fed into the custom analyzer that has been developed by extending Mahout Machine learning library. The StopWordsApplied is similar to the MySQLStopWords table by design.

The flow chart below sums up the entire stop words elimination process followed by the tool.

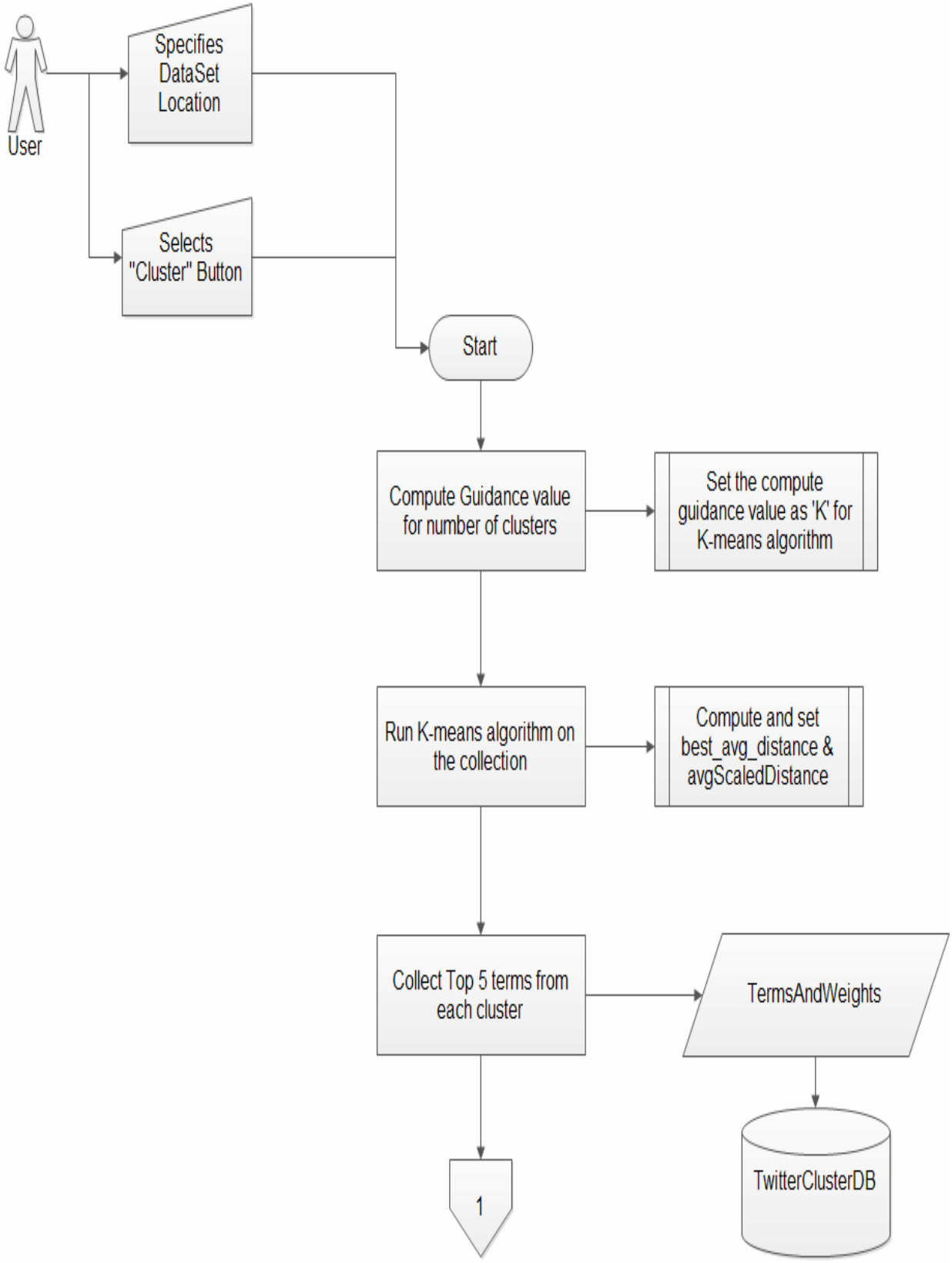

Figure 7: Stop Word elimination process part 1

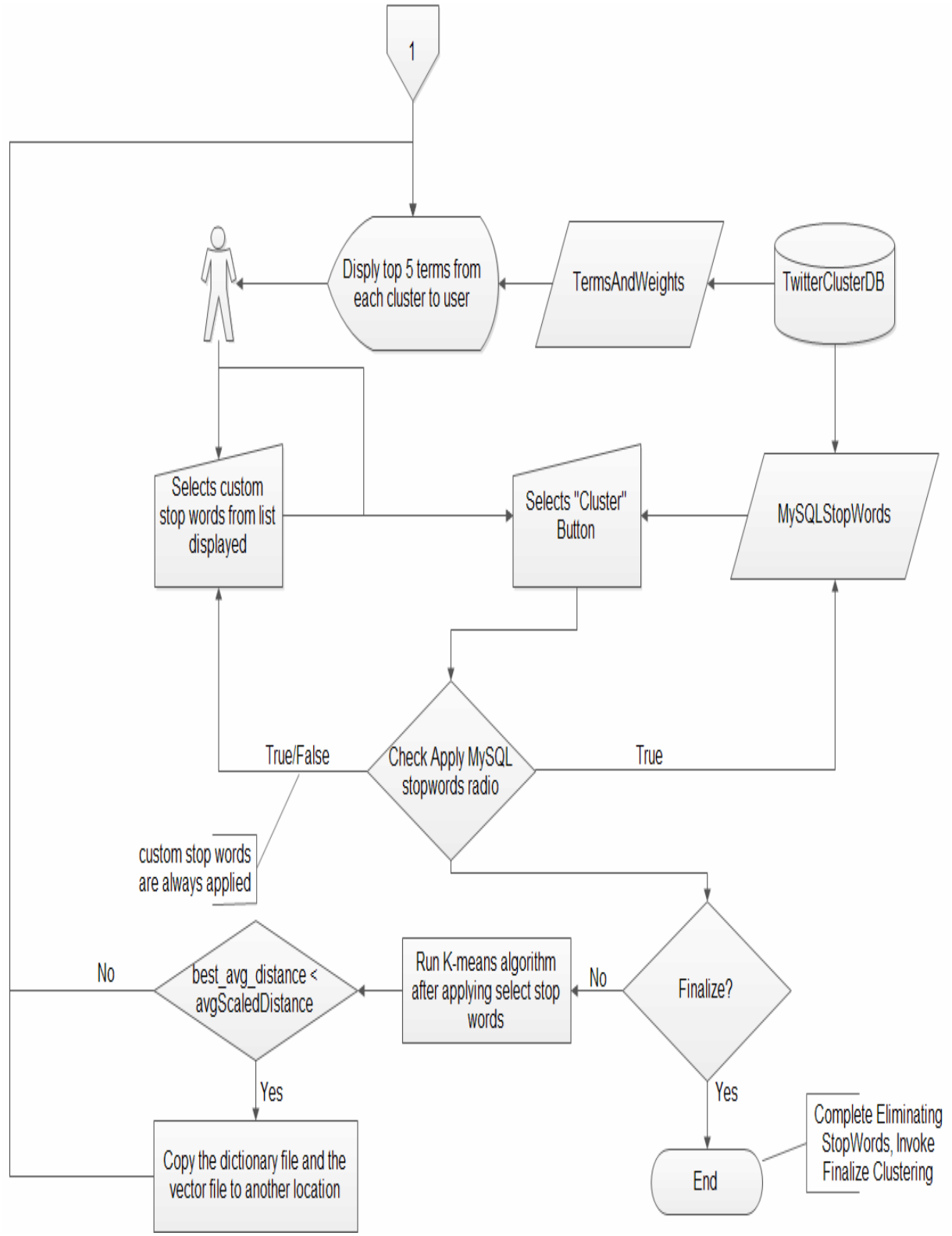

Figure 8: Stop Word elimination process part 2

As it can be seen from the flow chart, the tool saves the dictionary of best achieved quality of clustering on each run. This ensures that the terms which are essential for the clustering are not eliminated and the best possible quality of clustering for that collection is achieved.

Finalize clustering

The scaled average inter-cluster distance is used to compare the quality of clustering on each run. As mentioned earlier the tool maintains the best achieved quality of clustering. The stop word elimination process continues till either the scaled average inter-cluster distance remains unchanged or drops below previously achieved scores. The user may decide at this point to complete the clustering procedure and may choose to finalize clustering. The figure below shows the Finalize button that the tool provides to invoke finalize clustering procedure.

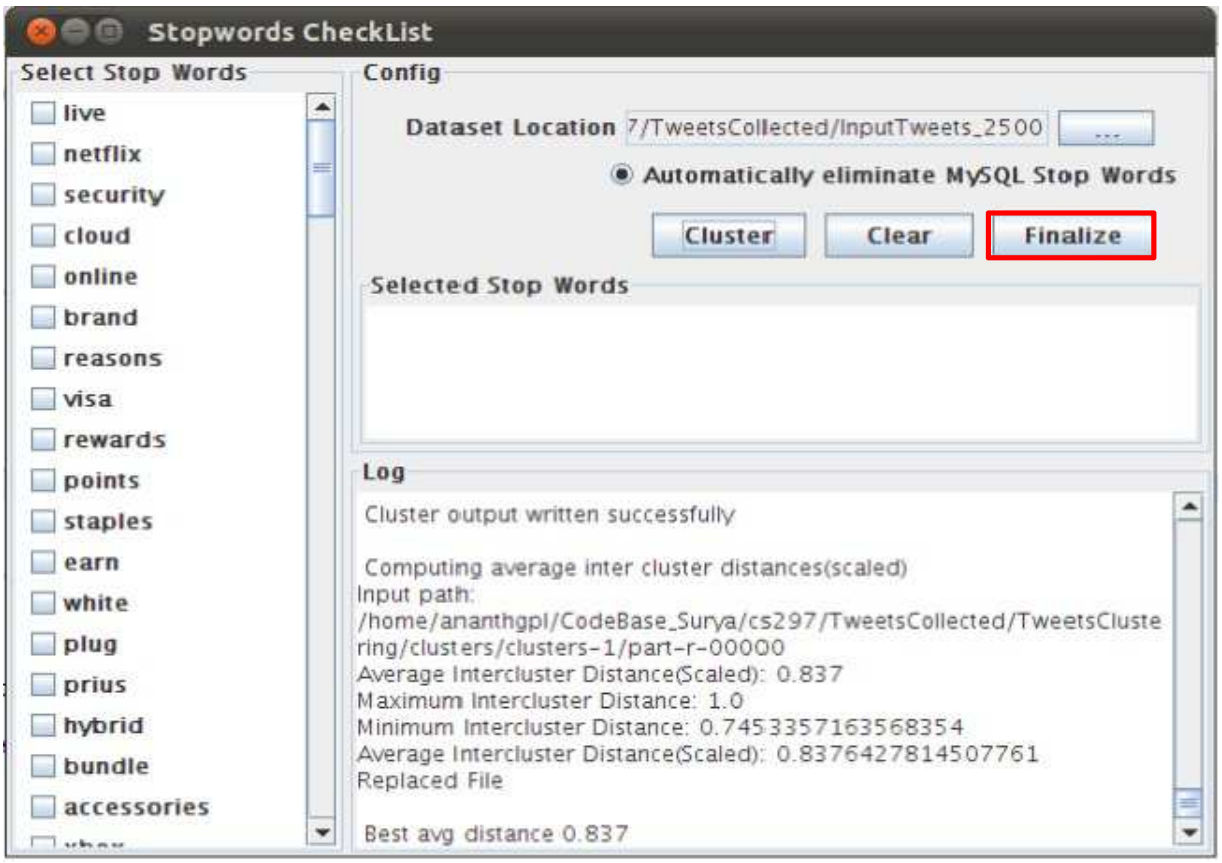

Figure 9: Finalize Clustering Button

The finalize procedure starts when the user clicks the highlighted button on the figure shown above. In the finalize procedure, the tool makes use of the best dictionary saved and the vector file that were saved previously from earlier runs of the clustering algorithm. The tool makes use of Canopy clustering algorithm during the finalization procedure to approximate the number of clusters. The following flow chart shows the overview of the finalize procedure:

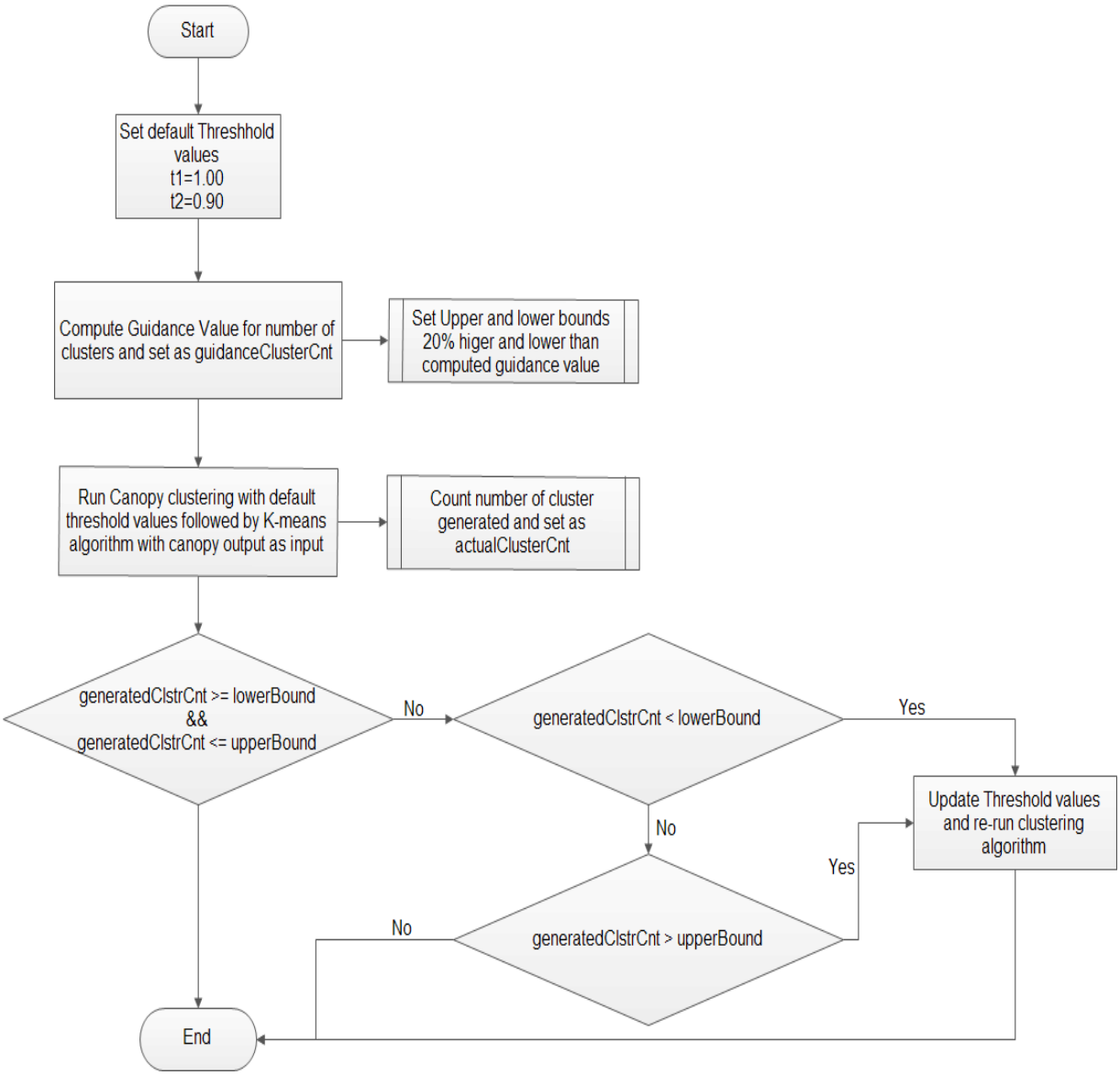

Figure 10: Overview of Finalize Clustering

The procedure shown in the figure clearly indicates that the clustering procedure can be finalized only when appropriate threshold values for canopy clustering is determined. The guidance value is used as a reference point to determine approximate number of clusters and the algorithm developed to finalize clustering aims to keep the number of clusters within 20 % range of the guidance value. However, in situations where the number of clusters approximated by canopy clustering exceeds the guidance value largely, the tool ignores the guidance value as estimated by canopy is far more reliable as compared to the guidance value computed.

The following pseudo-code sums up the finalize clustering procedure; the following sections will deal in detail with procedures to determine threshold & finalize clustering.

```
Finalize clustering overview:
Begin
t1 <- 1.00
t2 <- 0.90
k <- computeGuidanceClstrCnt(dataSetLocation)
upperBound <- (k + (0.20 * k))
lowerBound <- (k – (0.20 * k))
Execcute Canopy Clustering with t1 & t2 as threshold
actualClstrCnt <- countClusters()
If((actualClstrCnt < lowerBound)|| (actualClstrCnt > upperBound))
    update values of t1 & t2 and execute Canopy Clustering
EndIf
End
```

Canopy clustering

Canopy has been used by the tool to identify centroids approximately for the chosen dataset and for seeding the K-means algorithm. As discussed earlier the Mahout implementation of K-means makes use of RandomSeedGenerator class to determine initial centroids, which is fast but not a guaranteed technique to produce good estimates [2].

Canopy is a fast approximation technique which determines clusters in exactly one pass over the data [2]. The algorithm requires two threshold values to be provided as a part of the input. Consider the threshold to be T1 and T2 respectively and T1>T2 [2]. Beginning with an empty set of canopies the algorithm iterates over the data producing canopies. In each iteration the algorithm removes a point from the dataset and adds it to a canopy list as a center [2]. If the distance of point is within T1 the point is added to the list as a separate canopy, but if it is within T2 then it is removed from the list so that a new canopy is not formed. This process is repeated till all points in the list are exhausted [2]. Following command invokes canopy clustering algorithm from Mahout:

```
$Mahout_HOME/bin/mahout canopy
-i << Location of TF-IDF vectors>>
-o <<Location of output >> -dm << Distance Measure >>
-t1 <<T1>>
-t2 <<T2>>
-tempDir << Temporary directory to store intermediate canopies >>
```

Determine Canopy Thresholds and Finalize clustering

The finalize procedure by the tool primarily involves finding optimal thresholds for canopy clustering. The following flow chart shows in detail the procedure followed for the same:

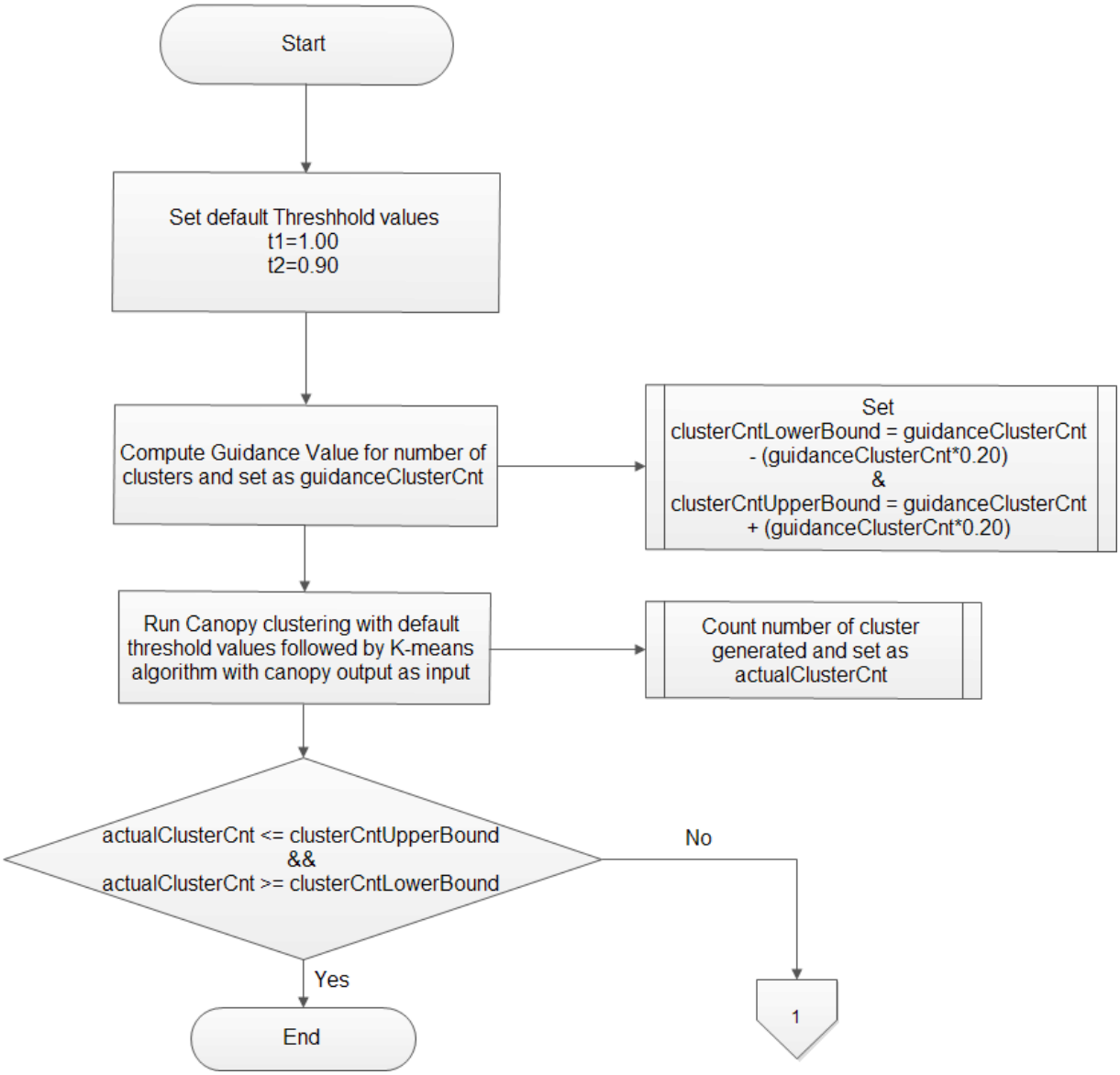

Figure 11: Determine threshold for Canopy Part I

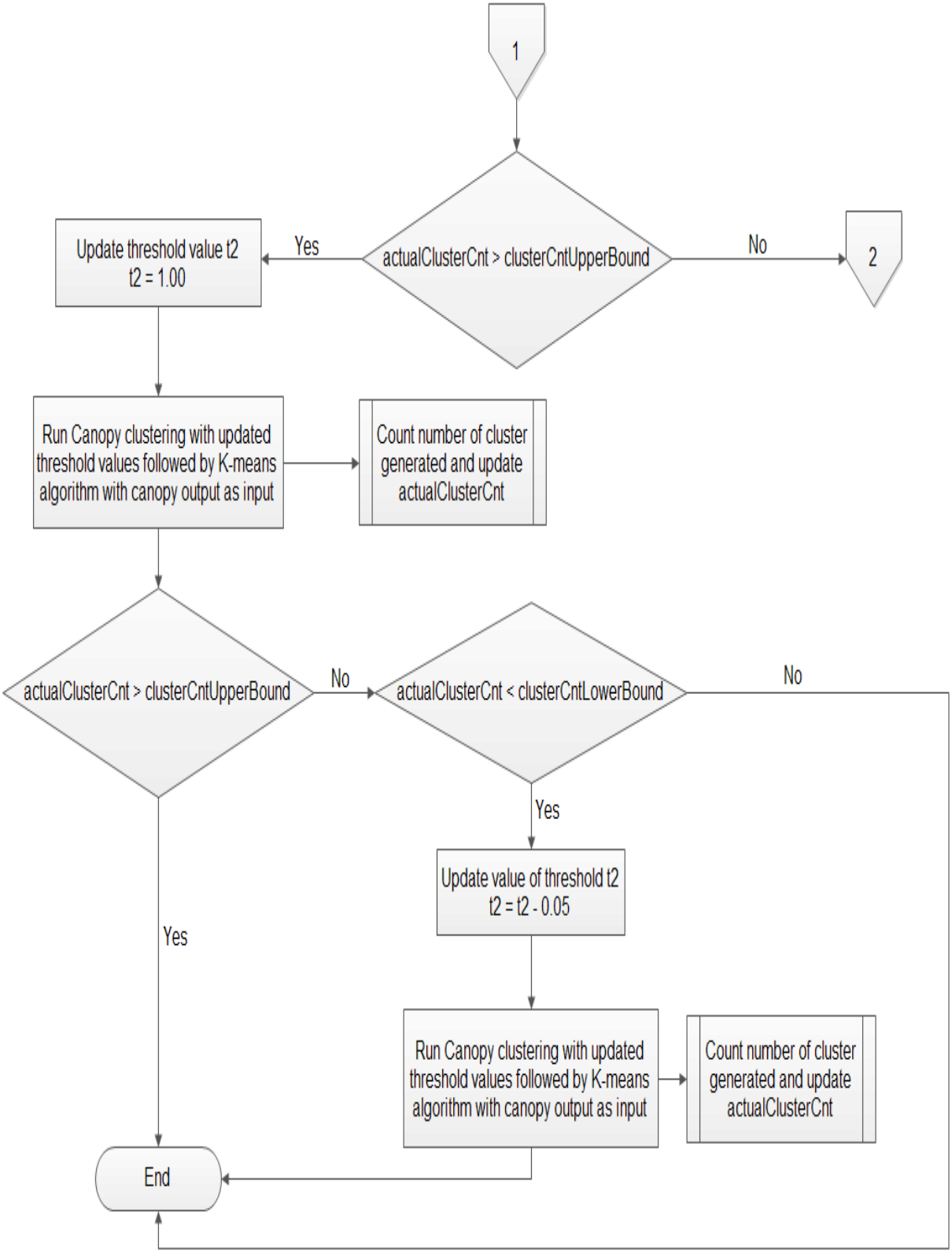

Figure 12: Determine Threshold For Canopy Part II

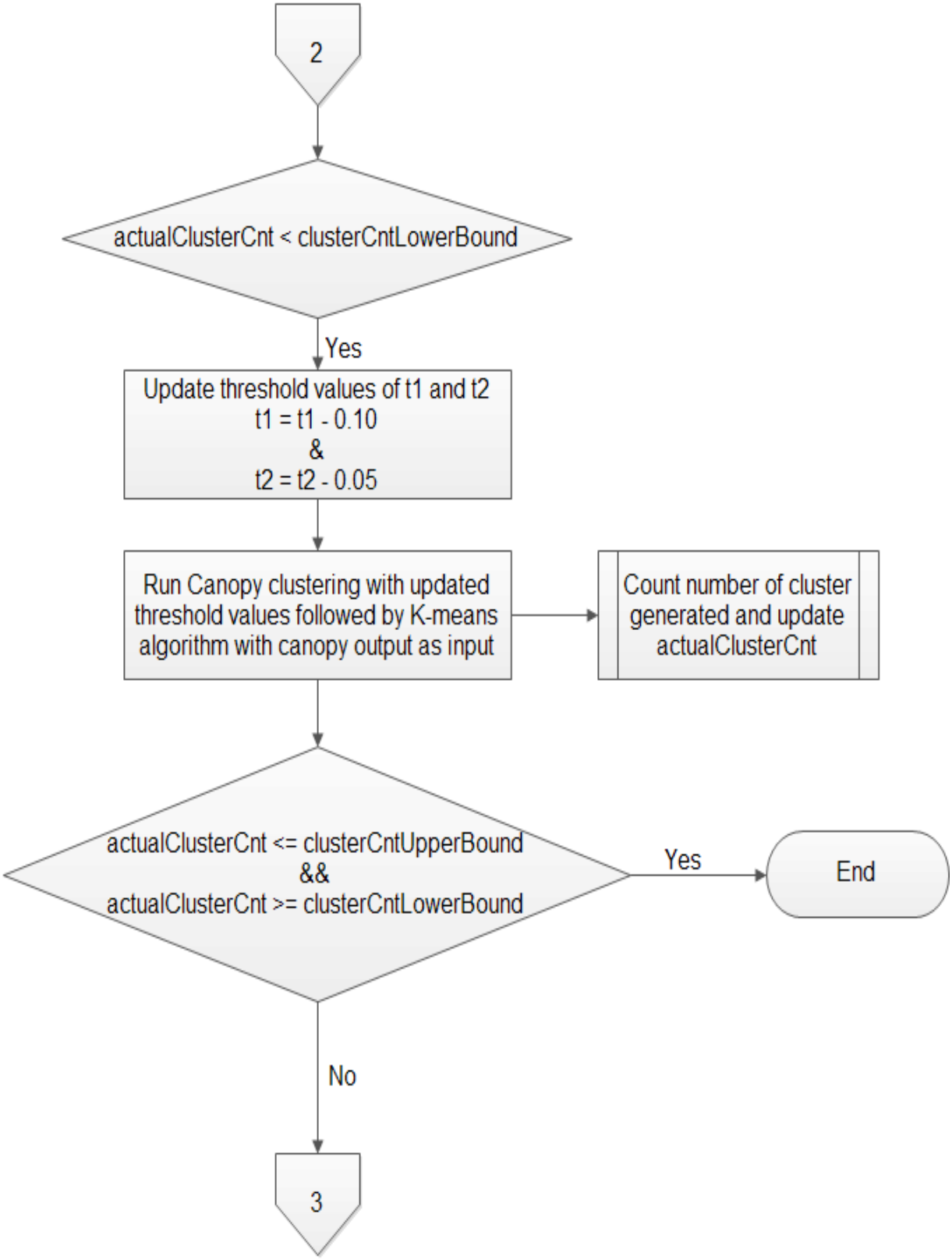

Figure 13: Determine Threshold for Canopy Part III

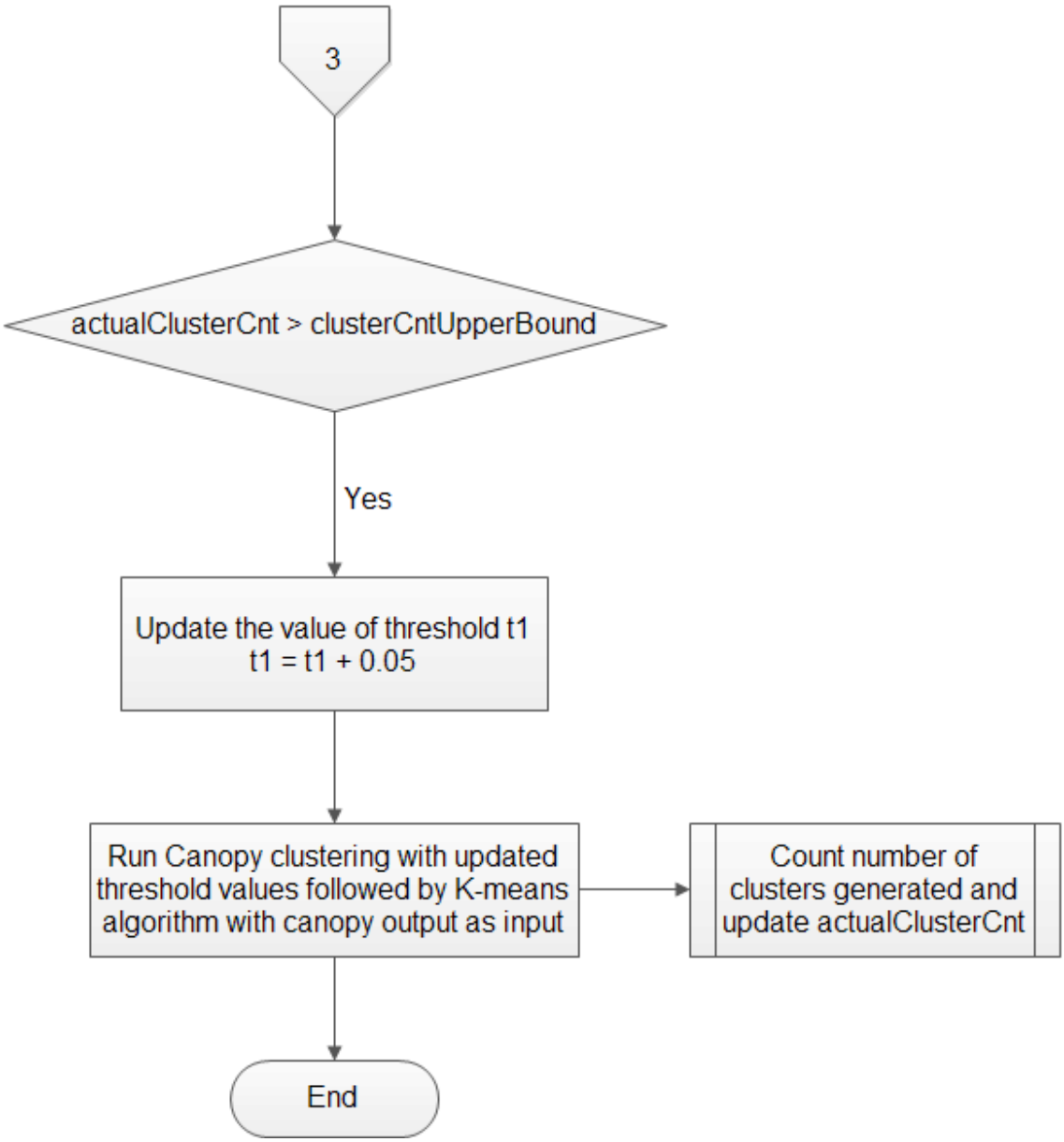

Figure 14: Determine Threshold For Canopy Part IV

The finalize procedure starts with computing a guidance value for the number of clusters. This guidance value shall be used as a reference to control the number of clusters that are produced. Based on observations on many runs, the initial values for the algorithm are T1=1.00

& T2=0.90. After one run of canopy clustering followed by K-means clustering, the tool counts the number of clusters generated.

If the number of clusters generated are within 20% higher or lower of the computed guidance value, then the clustering is completed and the program exits. The threshold values influence the number of clusters produced by the algorithm. If the generated cluster count is not within the specified range, then the guidance value is updated to increase or decrease the number of clusters produced.

If the number of clusters is greater than the specified range, the value of t2 is updated to 1.00 and the clustering algorithm is executed again. If the number of clusters at this time is still greater than the specified limit, then the clustering is finalized and the program exits as for cosine distance measure, the maximum value is 1.00. If the number of clusters drops below the specified range after the first update, then the value of T2 is updated to 0.05 less than original value of T2 and the programs executes clustering and exits.

If at the first run, the number of clusters is less than the guidance value computed, then value is T1 is reduced by 0.10 and T2 by 0.05 and clustering is run again. If the number of clusters is within specified range, clustering is finalized and the program exits, else if the number of clusters generated is greater than the guidance value computed, then the value of T1 is increased by 0.05. Clustering is performed again by the tool and the process is completed. The number of clusters produced at this time is the best estimate that the canopy clustering can generate with the specified thresholds.

If the number of clusters is still greater than the guidance value computed, then the tool finalizes the procedure with the generated clusters. The tool relies more on canopy clustering than on the guidance value as canopy clustering makes decisions based on the nature of the data. The guidance values have been used only to put a bound on the number of iterations that the algorithm should perform before concluding.

The following Pseudo-code sums up the process described above.

```
Determine Canopy Threshold & Finalize clustering:

Begin
t1 <- 1.00
t2 <- 0.90
k <- computeGuidanceClstrCnt(dataSetLocation)
upperBound <- k + (0.20 * k)
lowerBound <- k – (0.20 * k)
Execute Canopy Clustering with t1 & t2 as threshold
actualClstrCnt <- countClusters()
If((actualClstrCnt <= lowerBound)|| (actualClstrCnt >= upperBound))
    display actualClusterCnt and exit
else
    if(actualClstrCnt > upperBound)
       t2 <- 1.00
       execute Canopy Clustering with updated Thresholds
       actualClstrCnt <- countClusters()
       if(actualClusterCnt >= upperBound)
          display actualClstrCnt & exit
```

```
        else
           if(actualClstrCnt < lowerBound)
              t2 <- t2 - 0.05
              execute Canopy Clustering with updated thresholds
              display actualClstrCnt & exit
           EndIF
        EndIF
     else
     if(actualClstrCnt < lowerBound)
        t1 <- t1 – 0.10
        t2 <- t2 – 0.05
        execute Canopy Clustering with updated Thresholds
        actualClstrCnt <- countClusters()
        if((actualClusterCnt>=lowerBound)||(actualClusterCnt
>=upperBound))
           display actualClstrCnt & exit
        else
           if(actualClstrCnt > upperBound)
              t1 <- t1 + 0.05
              execute Canopy Clustering with updated thresholds
              display actualClstrCnt & exit
           EndIf
        EndIf
     EndIf
EndIf
End
```

Finally to present a summary of the entire clustering procedure by the tool starting from identifying dataset, identifying and eliminating stop words, determine appropriate thresholds for canopy clustering to the final execution of the clustering algorithm, Pseudo-code is shown below:

```
Overall Clustering Procedure:
dataSetLocation <- location selected by user
Begin
K <- computeGuidanceClstrCnt(dataSetLocation)
run k-means on dataset
|topTerms| <- collect top 5 terms from each cluster
|displayTerms| <- |topTerms| \ |mySQLStopWords|
While (¬finalize)
    CALL eliminateStopWords()
EndWhile
CALL finalizeClustering
End
```

Chapter 6

Third Party Tools

Third party tools have been used in developing and testing this tool. Apache Mahout, Apache Hadoop and Topsy are the main third party tools that have been used. This section presents the details of the third party tools used.

Apache Mahout

Apache Mahout is highly scalable machine learning tool and provides implementation for a number of algorithms [2]. Of these we have made use of the K-means and canopy algorithms apart from the utilities to prepare text data for clustering and to read the cluster output. The details of the usage of the tool has been covered in the earlier sections, refer chapter 5 for the same.

Apache Hadoop

Apache Hadoop is an open source scalable distributed computing library [5]. Hadoop has been used in a pseudo distributed mode for this tool. When Hadoop is used in this mode, the library just generates multiple java instances of the daemon and operates in a thread mode and exploits the multiple processors. These results in parallel processing of the large datasets by using map reduce algorithms.

Being highly scalable, large datasets are processed in parallel and reduces the execution time for the clustering algorithms on this tool.

Topsy

Topsy has been used in this paper to collect Tweets which are to be clustered. It is quite cumbersome to use the Twitter API directly as there are several restrictions on the number of API calls one can make in per day. To avoid this problem we make use of Topsy.com which maintains the Tweets and provides a java API to access them.

Twitter typically has a restriction on the duration for which Tweets are maintained per timeline. This time frame is typically very short. Also, only a specific number of Tweets are maintained per timeline. To resolve this issue some services have been archiving Tweets by streaming them daily and maintaining them in a database. One such service is offered by Topsy.com.

The API service that is offered is Otter Java API:

- Otter API is a RESTful HTTP web service to Topsy
- It provides access to Topsy search results, URL information, author influence
- An API Key can be obtained by signing up and providing brief details of intended usage
- Request Syntax in case of using an Http Request

  Host/Resource.Response?apikey=value**?QueryString=**search_query**?**Optional_Parameters
- For a registered user, each key is permitted 7000 calls per day

There are several options in terms of the resource to make use of. Based on our requirement the resource we have used in this application is */Search* which has the following properties:-

Name	Required/Optional	Description
q	Required	Query String for search
Window	Optional	Time window for results, example, dynamic, w, m, d, h
Type	Optional	tweets, image, video
List Parameters	Optional	Used for an advanced search

The required option ‘q’ has a specific syntax and ‘Window’ can be used to set the appropriate time frame required. As the focus in this application is on tweets alone, ‘type’ can be set to tweet to eliminate photos and videos [8].

It is not necessary that the Http request be used for collection of tweets. The Otter4j Java API that is offered has been used instead. The following are the features of the API which is still being updated:

- Finer control on the query parameters
- ListParameter is a POJO for providing a set of optional parameters

- The ListParameter will be extended to include a couple more parameters that is needed
- TopsyConfig is a class which can be used to specify API-key, so it is no longer necessary to include API Key in query each time explicitly
- SearchCriteria – Accepts required query attribute and ListParameter which contains optional parameters
- Post is another parameter that is offered which can be made use to access tweets directly rather than having to explicitly handle the JSON response

Personalize Otter API

The API could not be made use of as it is. It was necessary to make certain adjustments in the code based on observations and requirements. Since Otter Java API is an Open Source project its source code could be easily downloaded and analyzed. Following changes were made in order to suit the requirements of the application:-

1) Search method in Search.java

The core service offered by the API is its search functionality. In order to specify a specific time frame for the Tweets to be collected there are two options:

i) “Window”

ii) “mintime”&“maxtime”

Either of these options can be provided but not both to obtain a correct query syntax. However, in the search method in the API, it is seen that it checks for the option “Window” being null, but does not check whether “mintime” and “maxtime have been specified. This causes the “Window” to be set to “a” which is equivalent to entire time frame of two years. There a maximum of 1000 Tweets that can be retrieved as we cannot

get more than 10 pages each having 100 Tweets.

The work around for this problem was to extend search class and override the search function, and comment out the code that checks for “Window” being null and sets it to “a”. Below is the snapshot of the code that has been commented in the Search class in “search” method

2) fecthAll method in Seach.java

Another change that was made in this class was in the fetchAll method. The API would make 9 calls on behalf of the 1 call the user makes in the interest of returning maximum tweets per users call. However, considering that we have “mintime” and “maxtime” set for a shorter span of time, it is an overhead to make 10 calls, furthermore it is even more critical in case of a restriction on the number of calls that can be made per day.

There was a while loop in the fetchAll method that has been replace with a do While loop with necessary modifications to make not more than just enough calls to return all Tweets for specified time frame.

To illustrate this point, consider the case of there being a total of 225 Tweets for a specified time frame. The API would make 10 calls to return 225 Tweets, despite there being 100 Tweets in each page, thereby making 7 unnecessary calls. The new changes would check for the total number of Tweets, set count of Tweets per page to 100 and make every call only if previous call returned 100 Tweets, thereby making only 3 calls to return 225 Tweets in place of 10 calls to do the same.

3) toQuery method in SearchCriteria.java

It was noticed that there was a defect in the code which would cause the API to fail. The query would be built and appended here, hence causing the API to fail. This class too was extended and the toQuery method was overridden to fix the defect.

4) Convert Date to Unix Time

The mintime and maxtime options mentioned only accept unix time as arguments and hence a small utility method was written to achieve this.

Collecting Tweets

A utility to collect Tweets has been developed which has utility methods to set requisite parameters and make use of the response to write returned Tweets into a file. The utility consists of an array of identified user names whose Tweets we need to collect and a method which loops over the array setting requisite parameters and makes the API calls, finally appends the returned response to a file. This method makes the Tweet collection process a single step process:

➔ Set “mintime” and “maxtime” parameters and execute the code

The time frame for fixed for one month on each run, starting from 01/01/2010 to 12/31/2012. Once the Tweets have been collected, it is necessary to clean them and prepare them for clustering.

Sanitize Data for Clustering

Once the Tweets have been collected, it needs to be cleaned and prepared for clustering.

Special characters are eliminated, links from the tweets are eliminated and only pure text is retained. Finally, the file is split into different files, each containing one Tweet.

Complete Procedure

The complete procedure of acquiring Tweets to preparing them for clustering is given below:

1) Set time span in the getTweets method of TweetCollector.java
2) Execute the code to retrieve Tweets
3) Execute FilterTweets.java on the file in which Tweets have been collected
4) Finally execute FileSplitter.java on the output file of FilterTweets.java

Once these steps have been completed, a folder containing files with each file having a Tweet will be generated. The files in this folder can be used as an input directly to mahout.

Chapter 7

Experiments

The performance of the tool was compared in terms of the scaled average inter-cluster distance between the clusters generated by executing only the K-means algorithm as opposed to executing K-means algorithm after estimating number of clusters with the help of Canopy clustering. Eliminating the same set of stop words for different collections, may not work for all collections.

To compare the performance of the tool, the tool was run multiple times on different data sets. The quality of clustering obtained is compared to the quality of clusters obtained by the approach followed by *Surya Bhagavat and Teng Moh,* where only K-means algorithm is executed by providing the value of K to the algorithm and also eliminating a fixed set of stop words.

To determine the value of K, we make use of the Rule of Thumb as we execute on different collections of different sizes, but retain the same set of stop words that need to be eliminated for all the collections.

Tweets have been collected since 2010 through 2012. Lower bound and Upper bound columns are 20% lower and higher respectively as compared to the guidance value computed with the Rule of Thumb. The dataset size is the number of Tweets that have been clustered. The column K-means below is the result of executing only K-means as explained above, and the Tool column shows the quality after finalizing clustering on the Tool. The average of 10 runs is presented below.

For each year, the experiments have been carried and the result obtained is presented in the tables below:

Year 2010						
Data – Set Size	**No. Of Clusters (Rule of Thumb)**	**Lower Bound**	**Upper Bound**	**No. of Cluster (Generated by Tool)**	**Scaled Average Inter-Cluster Distance**	
					K-means	**Tool**
100	7	6	8	11	0.865	0.937
250	11	9	13	10	0.829	0.867
500	16	13	19	14	0.803	0.866
750	19	15	23	29	0.806	0.878
1000	22	18	26	20	0.830	0.913
2500	35	28	42	45	0.883	0.924
5000	50	40	60	95	0.908	0.943
7500	61	49	73	119	0.919	0.961
10000	71	57	85	169	0.928	0.972
19335	98	78	118	220	0.929	0.979

Year 2011						
Data – Set Size	**No. Of Clusters (Rule of Thumb)**	**Lower Bound**	**Upper Bound**	**No. of Cluster (Generated by Tool)**	**Scaled Average Inter-Cluster Distance**	
					K-means	**Tool**
100	7	6	8	15	0.803	0.960
250	11	9	13	17	0.840	0.859
500	16	13	19	14	0.788	0.900
750	19	15	23	29	0.884	0.906

1000	22	18	26	30	0.859	0.880
2500	35	28	42	61	0.872	0.916
5000	50	40	60	105	0.886	0.958
7500	61	49	73	120	0.899	0.966
10000	71	57	85	159	0.908	0.967
20000	98	78	118	301	0.920	0.982

Year 2012						
Data – Set Size	**No. Of Clusters (Rule of Thumb)**	**Lower Bound**	**Upper Bound**	**No. of Cluster (Generated by Tool)**	**Scaled Average Inter-Cluster Distance**	
					K-means	**Tool**
100	7	6	8	11	0.797	0.892
250	11	9	13	10	0.808	0.896
500	16	13	19	14	0.783	0.896
750	19	15	23	29	0.842	0.962
1000	22	18	26	20	0.891	0.932
2500	35	28	42	45	0.876	0.962
5000	50	40	60	95	0.894	0.941
7500	61	49	73	119	0.898	0.951
10000	71	57	85	169	0.908	0.972

The quality of clustering has improved when using the tool as compared to the quality obtained by just the K-means with a fixed set of stop words to be eliminated. The number of

clusters generated in many cases is far greater than the value estimated by the rule of thumb but the quality of clustering is better than when using the guidance count directly with K-means.

To test the performance of the new algorithm developed to identify threshold values for canopy and complete clustering as opposed to using only the k-means algorithm on the tool itself, the best average inter-cluster distance when eliminating stop words has been compared with the quality of clustering after finalizing clustering. The same data set with collections of different sizes has been used and the result has been presented in the table below:

Year 2010		
Data – Set Size	**Scaled Average Inter-Cluster Distance**	
	Before Finalizing (Only K-means on Tool)	**After Finalizing (Canopy followed by K-means Clustering)**
100	0.887	0.937
250	0.809	0.867
500	0.833	0.866
750	0.850	0.878
1000	0.890	0.913
2500	0.927	0.924
5000	0.939	0.943
7500	0.942	0.961
10000	0.940	0.972
19335	0.967	0.979

Year 2011		
Data – Set Size	**Scaled Average Inter-Cluster Distance**	
	Before Finalizing (Only K-means on Tool)	**After Finalizing (Canopy followed by K-means clustering)**
100	0.869	0.960
250	0.870	0.859
500	0.857	0.900
750	0.932	0.906
1000	0.859	0.880
2500	0.873	0.916
5000	0.890	0.958
7500	0.965	0.966
10000	0.931	0.967
20000	0.946	0.982

Year 2012		
Data – Set Size	**Scaled Average Inter-Cluster Distance**	
	Before Finalizing (Only K-means on Tool)	**After Finalizing (Canopy followed by K-means clustering)**
100	0.815	0.892
250	0.880	0.896
500	0.890	0.896
750	0.900	0.962

1000	0.887	0.932
2500	0.889	0.962
5000	0.926	0.941
7500	0.893	0.951
10000	0.938	0.972

These observations show that in very few cases the quality of clustering is better when K-means alone has been used. These may be attributed to anomalies due to the random seeding that K-means algorithm performs during clustering. This proves that K-means is slightly inconsistent as compared to the Canopy clustering algorithm that generates clusters very consistently provided the threshold values are chose appropriately. Multiple runs of canopy generation has also proved that the new algorithm developed to identified thresholds works well for technology Tweets.

Chapter 8

Conclusion

The tool described in this paper achieves two major goals: Automate the stop word elimination process and enhance quality of clustering of technology tweets as compared to the previous approach. However, it is observed that due to the nature of data, it is not possible to entirely automate the process as human intervention is required to identify acronyms and similar abbreviations. The tool uses an algorithmic approach to identify appropriate thresholds for Canopy clustering algorithm.

Results obtained from the experiments performed indicate that the quality of clustering improves if Canopy clustering is executed before K-means clustering and the stop words eliminated need to be specific to a collection as a fixed set of stop words do not work well for all collections.

Using the stop word elimination process by the tool and using Canopy clustering results in a notable improvement over making use of a manual approach. Also, the threshold identification algorithm has performed well with technology Tweets which can be seen by the improved quality of clusters as identifying appropriate threshold values is key in generating good clusters with canopy generation. It is also observed that the larger the size of the collection to be clustered the better quality of clustering is achieved by the tool.

Chapter 9

Future Works

One of the principal objectives of this paper was to automate the stop words elimination process. However, we have managed to semi-automate the process because of the presence of abbreviations in the text requiring human interaction. The use of a classifier to train the tool to identify and eliminate abbreviations needs to be evaluated as a part of extended research on this paper. This might yield a completely automated technique to stop word identification.

Further research is needed to improve the current new algorithm developed to identify appropriate thresholds for Canopy clustering. The current approach starts off with the use of fixed threshold values and modifies them based on the result of clustering. Instead, an approach similar to binary search can be evaluated to identify initial threshold values which can then be modified based on the quality of the clusters generated.

LIST OF REFERENCES

[1] Moh, Teng-Sheng, and Surya Bhagvat. "Clustering of technology tweets and the impact of stop words on clusters." *Proceedings of the 50th Annual Southeast Regional Conference*. ACM, 2012

[2] Mahout in Action
By Sean Owen, Robin Anil, Ted Dunning, Ellen Friedman
Publisher: Manning Publications
Released: October 14, 2011

[3] TF-IDF weights, Wikipedia, <http://en.wikipedia.org/wiki/Tf%E2%80%93idf>, N.p., n.d. Web. 03 Nov. 2012

[4] Apache Mahout, <https://cwiki.apache.org/confluence/display/MAHOUT/Overview>, N.p. n.d. Web. 13 Jan 2013

[5] Apache Hadoop, <http://hadoop.apache.org/>, N.p. n.d. Web. 13 Jan 2013

[6] "Introduction to data mining by Pang-Ning Tan, Michael Steinnbach, Vipin Kumar“

[7] Topsy Otter API, < http://code.google.com/p/otter4java/>, N.p. n.d. Web. 13 Jan 2013

[8] Otter API query specifics, <http://code.google.com/p/otterapi/wiki/Resources#/search>, N.p. n.d. Web. 13 Jan 2013

[9] "Apache Mahout: Scalable machine learning and data mining." Apache Mahout: Scalable machine learning and data mining. N.p., n.d. Web. 29 April. 2012. <http://mahout.apache.org>

[10] “Estimate Clusters Count”:: N.p., n.d. Web. 29 January 2013. < http://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set>

[11] Qing Chen; Shipper, Timothy; Khan, Latifur; , "Tweets mining using WIKIPEDIA and impurity cluster measurement," Intelligence and Security Informatics (ISI), 2010 IEEE International Conference on , vol., no., pp.141-143, 23-26 May 2010
doi: 10.1109/ISI.2010.5484758

[12] "Creating Vectors from Text." Mahout Wiki. N.p., n.d. Web. 30 Dec. 2011. <https://cwiki.apache.org/MAHOUT/creating-vectors-from-text.html#CreatingVectorsfromText-ConvertingdirectoryofdocumentstoSequenceFileformat>

[13] "AbstractVector (Mahout Math 0.6-SNAPSHOT API)." Search Lucene . N.p., n.d. Web. 30 Dec. 2011. <http://http://search-lucene.com/jd/mahout/math/org/apache/mahout/math/AbstractVector.html>

[14] "MySQL :: MySQL 5.6 Reference Manual :: 11.9.4 Full-Text Stopwords." MySQL :: Developer Zone. N.p., n.d. Web. 30 Nov. 2012. <http://dev.mysql.com/doc/refman/5.6/en/fulltext-stopwords.html>

[15] “Canopy Clustering” :: N.p., n.d. Web. 30 Nov. 2012 <https://cwiki.apache.org/confluence/display/MAHOUT/Canopy+Clustering>

[16] Swit Phuvipadawat and Tsuyoshi Murata. 2010. Breaking News Detection and Tracking in Twitter. In Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web

Intelligence and Intelligent Agent Technology - Volume 03 (WI-IAT '10), Vol. 3. IEEE Computer Society, Washington, DC, USA, 120-123. DOI=10.1109/WI-IAT.2010.205 <http://dx.doi.org/10.1109/WI-IAT.2010.205>

[17] Kanti Mardia et al. (1979). Multivariate Analysis. Academic Press.

[18] Purnami, Santi Wulan, Jasni Mohamad Zain, and Tutut Heriawan. "An alternative algorithm for classification large categorical dataset: k-mode clustering reduced support vector machine."

[19] Pham, D. T., S. S. Dimov, and C. D. Nguyen. "Selection of K in K-means clustering." *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 219.1 (2005): 103-119.

[20] Hamerly, Greg, and Charles Elkan. "Learning the k in A> means." *Advances in neural information processing systems* 16 (2004): 281.

[21] Cohen, William W., Pradeep Ravikumar, and Stephen E. Fienberg. "A comparison of string distance metrics for name-matching tasks." *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*. Vol. 47. 2003.

[22] Chen, Po Chih, and Teng-Sheng Moh. "Quality Improvement of Clustering Engine in the Internet Based on Correlation." *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE Computer Society, 2007.

[23] Zhang, D., Dong, Y., "Semantic, Hierarchical, Online Clustering of Web Search Results," *Proceedings of the 6th Asia Pacific Web Conference (APWEB)*, Hangzhou, China, Apr 2004, pp. 69-78

[24] Huang, Anna. "Similarity measures for text document clustering." *Proceedings of NZCSRSC* (2008): 49-56.