Short Text Summary Generation on Social Network Comment Streams

Pooja S. Choudhari\textsuperscript{A}, Sushma S. Nandgaonkar\textsuperscript{B}

\textsuperscript{A}Department of Computer Engineering, Savitribai Phule Pune University, Baramati, India
\textsuperscript{B}Department of Computer Engineering, Savitribai Phule Pune University, Baramati, India

Abstract

Recently, the popularity of social networking services has increased tremendously, so the amount of comments increases at a huge rate immediately when any post or message is generated by user. Thus, system focuses on the problem of obtaining short text summary on the comment set of a particular post on social network sites. The users of social sites always wish to obtain an overview of all comments instead of reading each and every comment in the set. So this system attempts to create groups of similar comments and provide brief summary for the particular post. As different users can demand the summary at any instant, existing clustering techniques cannot be used because they are unable to satisfy the real-time demand of such system. So the comment stream summarization problem in this paper is modelled as incremental clustering problem. This approach considers newly added comments in real time and thus provides clustering results in incremental manner. Finally, the visual interface is generated that help users to rapidly get an overview summary.

Keywords: Social networking, social posts, comment sets, incremental clustering, short text summary.

\section{I. INTRODUCTION}

Social networking has gained tremendous popularity in recent years as there are various social network sites available today which allows connecting people with one another from every corner of the world. Thus they serve as vital means of communication in our day-to-day life. On the largest social networking sites such as Facebook and Twitter, billions of interactions are generated every day which includes likes and comments. All such existing social platforms are very convenient to use and thus they attract large number of people to stay connected through them. Due to this reason, the celebrities, corporations, and organizations also create their own social pages to communicate with their fans and the other people. For each post, users can give their views by forwarding, giving a like, and leaving comments on it. Due to popularity of these platforms, not only the number of comments is more, but also the generation rate is remarkably high. Therefore users unnecessarily have to read all comments of each post and it is almost impossible every time. But still users wish to know what other people's are talking about and what are the views of these discussion participants.

Mostly, celebrities and corporations always desire to know how their fans and customers respond to their certain topics and content. Thus it has created the necessity to develop an enhanced process for summarizing comments on social network sites. Traditional comment series such as the discussion on products or movies generally express more complete information. But comment streams on social network sites are in short text style with a very general language used in daily life. Hence the main aim here is to distribute comments of one social post into different clusters. The group of comments having similar content are included in one cluster and in such a way different clusters get created for one social post.

This problem varies from existing processes and has different features and challenges. Firstly, the number of comments can grow at a high rate immediately when any social message is posted by any user. Also different users can request the summary at any instant. So an incremental method is necessary to satisfy the real time goals of this problem so as to quickly develop a summary based on the present comment set. Secondly, the comments on social network sites are usually short, and users mostly make use of informal and unstructured language and such language consist of acronyms, shortening words, etc. As a result, difficulty of finding the similarity between comments is more. Therefore instead of giving emphasis on the quality of clustering, the importance is given to develop a general summary instantly to give overview of a comment set to users.

This summarization problem can be modelled as a clustering task. However, existing clustering techniques cannot be used here due to their high computational complexity and also they do not meet the incremental need. The document clustering techniques which rely on topic modelling, such as latent dirichlet allocation (LDA) and latent semantic analysis (LSA),are not suitable here because of less information contained in every comment. Also these techniques prove to be inappropriate when there are small number of comments[1].

Thus the system in this paper makes use of incremental clustering task[1] which is used to create the clusters having various groups of views for social posts. For each cluster, important and common terms will be drawn out to form a key-term. Fully incremental algorithm is used that handles the outlier problem and also it can give clustering results with newly coming comments in real time. Thus the main objectives of this system are:

\begin{itemize}
  \item To develop different clusters for comments for each social message on basis of similarity of comments.
\end{itemize}
To incrementally update the clusters by considering newly arrived comments to meet its real time need.

To provide a short text summary to the social network users in the form of key terms so that they easily understand the comment sets.

II. RELATED WORK

A. Clustering Techniques

1) K-means clustering method[2]:

The division of M points is done in N dimensions to form K clusters. It helps to minimize the cluster sum of squares. A matrix of M points in N dimensions and a matrix of K initial cluster centres in N dimensions is needed as input. It searches K-partition with locally optimal within cluster sum of squares by transferring points from one cluster to other. It uses two algorithms- AS113(A transfer algorithm for non hierarchical classification) and AS58(Euclidean cluster analysis). These two algorithms require the time equal to number of iterations.

2) Topical Clustering of Tweets[3]:

This method is for automatically clustering and classifying twitter messages i.e. “tweets”, into different categories for eg. GoogleNews Service. Due to microblogging and social communication services, users post thousands of short messages every day. But keeping the track of all the messages posted by friends or other people is impossible & tedious. Unsupervised clustering used here applies LDA & K-means algorithm. Supervised clustering uses Rocchio classifier. Thus from each cluster, top few tweets are found to summarize a cluster.

B. Incremental Clustering techniques

1) Incremental K-means clustering algorithm[4]:

K-means clustering is computationally efficient, but faces the problem of converging at a local minimum. Thus a cluster centre jumping operation is used here which allows cluster centre’s to move in a radical way so that the overall cluster distortion is reduced. But, this method is very sensitive to errors. Addition of cluster centre’s is done one by one as they are formed. Compared to K means, this algorithm needs K-times more iterations because its number of iterations is equal to K & every iteration is equal to one execution of K-means algorithm.

C. Summarization techniques

1) Summarizing user contributed comments[5]:

It is used for summarizing user contributed comments by users on social web. The goal is to select the most representative comments from a large collection of user contributed comments. From the set of n-user contributed comments, best top-k comments are selected for summarization using two approaches first, clustering based approach(K-means & LDA) which identifies correlated groups of comments and second, a precedence based ranking framework which automatically selects informative user contributed comments. As precedence ranking is combined with topical clustering, this system yields overall higher performance compared to traditional document summarization methods.

2) IMASS: Intelligent Microblog Analysis and Summarization System[6]:

It is used to summarize a microblog post & its responses. It helps the readers to get a more constructive set of information in a efficient manner. It is a two step process. First, the post plus and its responses are classified into four categories based on the intention, interrogation, sharing, discussion and chat. Second phase uses different strategies like opinion analysis, response pair identification and response relevancy detection, to summarize a post & display critical information. Microblogs have different characteristics in terms of length & writing skills than other online information sources as news articles. Thus this scheme gives an effective strategy to summarize post based on its intention & type.

III. SHORT TEXT SUMMARIZATION SYSTEM

This system gives fully incremental algorithm[1] for “Short Text Summarization(STS)”. Incremental & BatchSTS[1] clustering algorithms used here to generate comment clusters, slightly sacrifices cluster quality as compared to the existing methods, but satisfy the real-time processing need of comment stream summarization task. The main difference between existing clustering methods & this system is that it maintains the radius for every cluster below a predefined threshold. The system architecture is as shown in Fig.1. It works in two phases as Clustering Phase & Summarization Phase.

A. Pre-processing of Comment streams

In preprocessing, removal of stopwords, nonwords and stemming is performed. First for each word in a comment it is checked that whether it is a stopword. If it is stopword then it is eliminated otherwise that word is sent to the nonword removal procedure. Then finally the words obtained after this procedure are reduced to their stem form by using “Standard Porter Stemming Algorithm”[7]. For eg. the derived word “making” gets converted into its root form “make” due to the stemming.

B. Term Vector Model Representation

The term vector model converts every comment into a set of terms. Term vectors for comments \( v_a \) and \( v_b \) are represented in the term vector model as:

\[
 v_a = (t_{1,a}, t_{2,a}, \ldots, t_{N,a}) \quad \text{and} \quad v_b = (t_{1,b}, t_{2,b}, \ldots, t_{N,b}).
\]

Here N is the dimension. The term \( t_{i,a} \) is the count of occurrence of term \( t_i \) in comment \( v_a \). As we represent comments in term vectors, so we adopt cosine similarity as a comment similarity measure which is given as:

\[
 \text{cosSim}(v_a, v_b) = \frac{\sum (t_{i,a} \cdot t_{i,b})}{\sqrt{\sum (t_{i,a})^2} \cdot \sqrt{\sum (t_{i,b})^2}} \quad (1)
\]

This eq(1) also helps to find cluster similarity.
C. Clustering Phase

1) BatchSTS Clustering Procedure[1]

This is the initial clustering phase for the formation of clusters of given comment set of any particular post as shown in Algorithm 1. The whole comment set $S$ and the radius threshold $k$ are the two inputs used here $k$ is used to restrict the number of comments in one cluster. There are two stages in BatchSTS as:

a) In steps 1-10 of Algorithm 1, first a cluster set $C$ is defined to store all the clusters. Then initial cluster $c_0$ is formed by adding 1st comment $S_0$ in $S$. For $S_j$ remaining comments, $sim(r, c_j)$ is evaluated using eq(1). $sim(r, c_j)$ gives similarity of comment $S_j$ with all clusters in $C$. If there exists any cluster $c_j$ whose similarity with $S_j$ is greater than or equal to predefined similarity threshold $mThr$ then that cluster is assigned as matched cluster and its similarity value is temporarily stored in some variable to check whether the next cluster has more similarity with $S_j$ than previous cluster. Finally match gives the most matching cluster having greater similarity with $S_j$ than all other clusters in $C$.

b) The second stage includes the checking of radius threshold of most matching cluster obtained in stage 1, as shown in steps 11-15. If its radius is less than predefined threshold $k$, then only the comment $S_j$ is added to $c_{match}$. Otherwise new cluster is created for it. If there is no any matching cluster in $C$ for comment $S_j$ then also a new cluster is created for it as shown in step 17. Finally set $C$ gives all the clusters formed using BatchSTS algorithm.

2) Incremental Clustering Procedure

In the second clustering phase, IncreSTS algorithm in [1] is modified as per the requirements. Incremental clustering is implemented as shown in Algorithm 2. It is used mainly to consider newly arriving comments in real time & immediately adding it into available BatchSTS clusters using similarity criteria. It works in two stages as:

a) In steps 1-8, the similarity of newly arrived comment $v$ with all the clusters in the output set $C$ of Algorithm 1 is calculated by $sim(v, c_j)$ using eq(1). These similarity values are added to set $sims$ & respective clusters are added to set newClusters. Then these clusters are sorted according to similarity values in descending order so that more similar clusters will be at the top. The newClusters set is updated by storing the sorted clusters.

b) In steps 9-15, a proper cluster is found among the sorted clusters, to add comment $v$. If a cluster having radius below threshold $k$ & similarity above similarity threshold is found, $v$ is added into it. Otherwise new cluster is created for $v$.

---

Algorithm 1 BatchSTS Algorithm[1]

```
Input : Previous clustering result set $C$
Output: Cluster set $C$
1. Set similarity threshold $mThr$
2. Initialize cluster set $C$
3. Form first cluster $c_0$ with first comment in $S_0$
4. Add $c_0$ to $C$
5. for each remaining comment $S_j$ in $S$
6.    Initialize $flag = 0$ & $match = -1$
7.    for each cluster $c_j$ in set $C$
8.        if $sim(r, c_j) >= mThr$ && $sim >= flag$
9.            Assign $sim$ value to $flag$
10.           Assign $j$ value to $match$
11.         if cluster $c_{match}$ exceeds radius threshold
12.            Create new cluster $nc_j$ for comment $S_j$
13.            Add $nc_j$ to $C$
14.        else
15.            Add $S_j$ to most matching cluster $c_{match}$
16.            Add $nc_j$ for $S_j$ & add it to $C$
17.            Create new cluster $nc_j$ for $S_j$ & add it to $C$
18. return $C$
```

Algorithm 2 IncreSTS Algorithm

```
Input : Previous clustering result set $C$
Newly incoming comment $v$
Radius threshold $k$
Output: Update Cluster set $C$
1. Initialize set $sims$ for $sim$ values
2. Initialize cluster set newClusters
3. for each cluster $c_j$ in set $C$
4.    Calculate $sim(v, c_j)$
5.    Add $sim(v, c_j)$ to set $sims$
6.    Add cluster $c_j$ to set newClusters
7. Sort the clusters by sortClusters(newClusters, sims)
8. Add sorted clusters to set newClusters
9. for each cluster $c_j$ in set newClusters
10.    if $rad(c_j) < k$ && $sims(c_j) >= simThr$
11.        Add new comment $v$ into $c_j$
12.    else
13.        Create new cluster $nc_j$ for $v$
14.        Add ‘nc_j’ to set newClusters
15. return newClusters
```
D. Summarization Phase

1) Formation of Concept:

Similar clusters are merged into one concept. We propose to summarize a concept instead of a cluster because it gives better knowledge for summary generation.

Initially, first concept is created by adding 1st cluster in set C returned by Algorithm 2. For each remaining cluster in C, it is checked whether it matches with available generated concepts. The cluster is added into concept with which it has greater similarity value. Otherwise new concept is created for it. Likewise, set of concepts is generated. Concepts having very less number of comments are removed as a noise, as they are not so helpful while generating the summary.

Algorithm 3 Summarization Algorithm

Input : nGrams and concept Comments
Output: Set of Key Terms as summary

1. Initialize list nGrams for nGrams of each concept
2. for each concept in set of concepts
3. Collect unigram, bigrams & ngrams
4. Add these grams to list nGrams
5. Apply removeNoise(nGrams)
6. Add updated grams to list nGrams
7. Set distribution threshold distThr
8. for each gram in set of nGrams
9. Initialize gCount=0 and dist=0
10. for each comment in cComments set
11. if comment contains gram
12. Increment gCount by 1
13. dist = gCount/size of cComments
14. Add dist value to list dists
15. for each dist value in dists
16. if dist(d) >= distThr
17. Add gram(d) to list newGrams
18. for each gram in newGrams
19. Add gram to set summary
20. return summary

2) Summarization procedure:

The Key-term extraction algorithm in [1] is modified as per the need of summary generation as shown in Algorithm 3. Comment set & set of nGrams of one concept are given as an input. It works in three stages as: It works in three stages as:

a) Initially in steps 1-6, unigrams, bigrams and nGrams are generated for every concept. Here we consider nGrams upto 3-grams. After this, repeated or duplicate grams are removed as a noise. Following example illustrates the removal of noisy grams:

Example: Consider the grams such as ‘walt’, ‘disney’, ‘walt disney’. As the first 2 unigrams are present in bigram ‘walt disney’, so these two unigrams will be removed as duplicates.

b) In steps 7-14, distribution of every gram in its concept is calculated. First the occurrence of every gram in its concept comments(cComments) is evaluated, i.e. in how many comments that gram is present is determined and this count is stored in variable gCount for finding distribution. Then the ratio of gCount to number of cComments gives distribution of a gram in its respective concept.

c) Finally in steps 15-20 only those grams are taken whose distribution is greater than or equal to the distribution threshold. Such grams together form key terms of a particular concept & are returned as a summary.

E. Event or Action Detection:

We assume an action in a comment as an event which occurred during the discussion on a post. We apply Stanford NLP Parser⁠[ on each comment for detecting the Part of Speech(POS) and then collect the grammatical tags of each word to model event or action. Finally the chain of interesting events is generated for particular comment list.

Parts Of Speech(POS) tagging process assigns one of the parts of speech to each word such as Verb(VB), Verb past tense(VBD), Noun(NN), etc.

Algorithm 4 Event Detection Algorithm

Input : Comment set of one post
Output: Events/Action

1. for each comment in post do
2. event, action = Ф
3. Apply POS(comment)
4. for each tag in POS do
5. if tag.equals(VB|VBD|VBN)
6. for each remaining tag in POS do
7. action+=tag
8. end if
9. end for
10. event = event ∪ action
11. end for

IV. EXPERIMENTAL RESULTS

A. Dataset:

We have focused on comment streams on Facebook, as its comments are in short text style and in general language. Real comment streams are collected from some selected Facebook pages i.e the pages with more number of likes. From each page, some social messages i.e posts are collected along with their respective comment set. Table1 gives the information about number of post and comments for some entities of our dataset.

B. Outcome:

We tested performance of this system by using dataset...
including about 670 comments of some entities on Facebook shown in Table I. The number of clusters generated are also shown in Table I. The radius threshold \(k\) is set to 3.

The thresholds \(m_{Thr}\) & \(sim_{Thr}\) in BatchSTS & Incremental algorithm respectively are set to 0.5 which is \(min\) threshold value for cosine similarity given by eq(1). The \(dist_{Thr}\) used in Algorithm 3 is set to 0.3. The summary generated for one concept is composed of different terms. It is observed that for higher values of \(dist_{Thr}\), less number of terms are generated which results in improper summary results. This shows that the value of \(dist_{Thr}\) is inversely proportional to number of terms in the summary.

<table>
<thead>
<tr>
<th>Posts</th>
<th>Comments</th>
<th>Retrieved Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BatchSTS</td>
</tr>
<tr>
<td>10</td>
<td>307</td>
<td>207</td>
</tr>
<tr>
<td>10</td>
<td>246</td>
<td>148</td>
</tr>
<tr>
<td>04</td>
<td>116</td>
<td>73</td>
</tr>
</tbody>
</table>

“Fig. 2” shows that the number of clusters generated using BatchSTS method goes on decreasing as radius threshold \(k\) increases and vice-versa. Thus it can be said that if \(k\) is set to a greater value then time required for clustering is less. But the greater \(k\) value is more suitable in the case of dense data. In case of sparse data, \(k\) should be small to preserve cluster similarity, but this increases time complexity of clustering process.

“Fig. 3(a)” shows the number of clusters generated by BatchSTS method depending upon number of input comments, similarity between those comments & radius threshold value \(k\).

“Fig. 3(b)” shows the number of clusters generated by Incremental method. In 1st case, number of comments are increased as compared to 1st case in BatchSTS, due to the addition of new comments. But the number of clusters has remained same, as all the newly added comments were similar to the previous clusters. So only the available clusters are updated. In 2nd & 3rd cases, the number of clusters have increased because of two reasons as: not all of the newly added comments are similar to the available clusters or newly arrived comment is similar with some cluster but cannot be added into it if its addition exceeds threshold \(k\). Then in these two situations, new clusters are formed. Thus it seems that, here number of clusters generated depends on three factors as, number of newly added comments, their similarity with available BatchSTS clusters & radius threshold value \(k\).

**V. CONCLUSION**

Thus, it seems that, for real time comment stream summarization problem on social networking sites, incremental clustering technique proves to be useful. Incremental clustering procedure consider newly incoming comments, so the clusters are obtained in incremental manner. These clusters are then summarized so that users can get an overview of all comments easily and rapidly instead of reading the set of comments of each post. Hence user obtains updated summary at any instant. Existing clustering methods which are presented here mainly focus on maintaining cluster quality hence they cannot provide real time updated summary of comments. But the technique in this paper satisfy this goal of obtaining incremental summary of comment sets on various posts, on social network sites and hence reduces the user’s efforts. In future, use of language semantics can be considered to give more appropriate summary results.

**ACKNOWLEDGMENT**

We express our gratitude towards Vidya Pratishthan’s Kamalnayan Bajaj Institute of Engineering and
Technology, Baramati for providing encouraging environment for carrying this research work.

REFERENCES


