Small Line Recapped Propagation On Social Network Illustration Courses

N.Vijay Kumar, U.Veeresh
Assistant Professor, CSE Dept., CMRIT
Hyderabad, India

Abstract- Recently, the popularity of social networking services has increased terrifically, so the amount of comments increases at a huge rate right away when any post or message is generated by user. Thus, system focuses on the problem of obtaining small line recapped 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 as an alternative of reading each and every comment in the set. So this system attempts to create groups of similar comments and provide brief review for the particular post. As different users can demand the recapped at any second, existing clustering techniques cannot be used because they are unable to gratify the instantaneous demand of such system. So the comment brook representation problem in this dissertation is modeled as incremental clustering difficulty. This loom considers a moment ago added comments in genuine time and thus provides clustering results in incremental manner. as a final point, the illustration crossing point is generated that help users to hurriedly get an general idea recapped.

Keywords- Social networking, social posts, comment sets, incremental clustering, small line recapped.

I. INTRODUCTION

Social networking has gained fantastic fame in modern years as there are various social network sites available today which allows connecting people with one another from each place of the planet. Thus they serve as vital means of announcement in our daily life. On the biggest social networking sites such as Face book and Twitter, billions of communications are generated every day which includes likes and comments. All such accessible social platforms are very expedient to use and thus they be a focus for huge number of citizens to stay connected through them. Suitable to this motive, the celebrities, corporations, and organizations also create their own common pages to communicate with their fans and the other people. For each job, users can give their views by forwarding, giving a like, and send-off comments on it. outstanding to reputation of these platforms, not only the number of comments is more, but also the proliferation rate is remarkably high. Therefore users gratuitously have to read all comments of each post and it is almost impracticable every time. But still users wish to know what other people are discussions about and what the views of these conversation participants are.

Habitually, celebrities and corporations always aspiration to know how their fans and customers act in response to their convinced topics and content. Thus it has created the necessity to develop an improved process for review comments on social network sites. Habitual comment series such as the discussion on products or movies generally express more complete information. But graphic 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 dispense comments of one social post into different clusters. The groups of illustrations having similar content are included in one cluster and in such a way dissimilar clusters get created for one social post.

This problem varies from existing processes and has different features and challenges. Firstly, the number of illustrations can grow at a high rate immediately when any social message is posted by any user. Also different users can request the recapped at any instant. So an incremental method is necessary to satisfy the real time goals of this problem so as to quickly develop a recapped based on the present illustration set. Secondly, the illustrations 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 illustrations is more. Therefore instead of giving emphasis on the quality of clustering, the importance is given to develop a general recapped instantly to give overview of a comment set to users.

This representation problem can be modeled 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 modeling, such as latent dirichlet allocation (LDA) and latent semantic analysis (LSA),are not suitable here because of less information contained in every illustration. Also these techniques prove to be inappropriate when there are small numbers of illustrations [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 illustrations in real time.

II. RELATED WORK

A. Clustering Techniques

1) K-means clustering process[2]:

The separation of M points is complete in N dimensions to form K clusters. It helps to reduce the cluster sum of squares. A medium of M points in N dimensions and a medium of K first cluster centers in N dimensions are required as input. It searches K-partition with locally best within cluster sum of squares by transferring points from one cluster to other.

2) Topical Clustering of Tweets [3]:

This method is for mechanically clustering and classifying twitter communication i.e. "tweets", into different categories for e.g. Google News Service. Due to micro blogging and social communication services, users job thousands of short messages every day. But keeping the path of all the messages posted by friends or other people is unfeasible & wearisome. Unsupervised clustering used here applies LDA & K-means algorithm. Supervised clustering uses classifier. Thus from each cluster, peak few tweets are found to recapitulate a cluster.

B. Incremental Clustering techniques

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

K-means clustering is computationally well-organized, but faces the difficulty of converging at a local smallest amount. Thus a cluster centre jumping process 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 centres is done one by one as they are formed. Compared to K means, this algorithm needs K-times more iteration because its number of iterations is equal to K & every iteration is equal to one execution of K-means algorithm.

C. Representation Techniques

1) Review user contributed comments[5]:

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

2) IMASS: Intelligent Micro blog Analysis and Representation System [6]:

It is used to summarize a micro blog post & its responses. It helps the readers to get a more constructive set of information in an 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. Micro blogs has 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 REPRESENTATION SYSTEM

This system gives fully incremental algorithm [1] for "Short Text representation (STR)". Incremental & BatchSTS [1] clustering algorithms used here to generate illustration clusters; slightly sacrifices cluster quality as compared to the existing methods, but satisfy the real-time processing need of illustration stream representation 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 & Representation Phase.

A. Pre-Processing of illustration streams

In preprocessing, removal of stop words, nonwords and stemming is performed. First for each word in a comment it is checked that whether it is a stop word. If it is stop word 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 e.g. 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 illustration into a set of terms. Term vectors for illustrations v_a and v_b are represented in the term vector model as:

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

Here N is the dimension. The term $t_{i,a}$ is the count of occurrence of term ti in comment va. As we represent illustrations in term vectors, so we adopt cosine similarity as a comment similarity measure which is given as:

$$\text{cossim}(v_a, v_b) = \frac{\Sigma(t_{i_a} * t_{j_a})}{\sqrt{\Sigma(t_{i_a})^2} \cdot \sqrt{\Sigma(t_{j_a})^2}} \qquad (1)$$

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

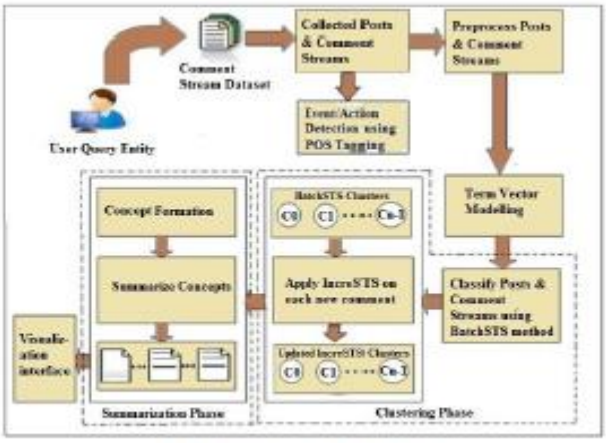

Figure 1. System Architecture

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 illustration set S and the radius threshold k are the two inputs used here k is used to restrict the number of illustrations 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 illustration S_0 in S. For S_r remaining illustrations, sim(r, c_j) is evaluated using eq (1). Sim(r, c_j) gives similarity of illustration S_r with all clusters in C. If there exists any cluster c_j whose similarity with S_r 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_r than previous cluster. Finally match gives the most matching cluster having greater similarity with S_r 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 illustration S_r is added to c_{match}. Otherwise new cluster is created for it. If there is no any matching cluster in *C* for illustration S_r then also a new cluster is created for it as shown in step 17. Finally set *C* gives the entire 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 illustrations in real time & immediately adding it into available BatchSTS clusters using similarity criteria. It works in two stages as:

Algorithm 1 BatchSTS Algorithm [1]

Input: Comment set *S* and Radius threshold *k*
Output: Cluster set *C*

1. Set similarity threshold *mThr*
2. Initialize cluster set *C*
3. Form first cluster c_0 with first illustration in S_0
4. Add c_0 to *C*
5. for each remaining illustration S_0 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_n for illustration S_r
13. Add nc_n to *C*
14. Else
15. Add S_r to most matching cluster c_{match}
16. Else
17. Create new cluster nc_n for S_r & add it to *C*
18. return *C*

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 *new Clusters*. Then these clusters are sorted according to similarity values in descending order so that more similar clusters will be at the top. The *new Clusters* set is updated by storing the sorted clusters.

b) In steps 9-15, a proper cluster is found among the sorted clusters, to add illustration *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 2 IncreSTS Algorithm

Input: Previous clustering result set *C*
Newly incoming illustration *v*
Radius threshold *k*
Output: Update Cluster set *C*

1. Initialize set *sims* for *sim* values
2. Initialize cluster set *new Clusters*
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 *new Clusters*
7. Sort the clusters by sort Clusters (*new Clusters*, *sims*)
8. Add sorted clusters to set *new Clusters*
9. for each cluster c_j in set *new Clusters*
10. if *rad(c_j) < k && sims(c_j) >= simThr*
11. Add new illustration *v* into c_j
12. else
13. Create new cluster nc_n for *v*
14. Add „nc_n" to set *new Clusters*

D. *return new Clusters 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 illustrations 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) Representation procedure:

The Key-term extraction algorithm in [1] is modified as per the need of recapped propagation as shown in Algorithm 3. Illustration set & set of nGrams of one concept are given as a 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 illustrations (cComments) is evaluated, i.e. in how many illustrations 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.

A Event or Action Detection:

We assume an action in an illustration as an event which occurred during the discussion on a post. We apply Stanford NLP Parser[1] on each illustration 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 illustration 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 =Φ
3. Apply POS(comment)
4. for each tag in POS do
5. if tag.equals(VB|VBD|VBN)
6. while(tag+1.equals(NN|DT|IN|PRP))
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 illustration streams on Face book, as its comments are in short text style and in general language. Real comment streams are collected from some selected Face book 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 illustration set. Table1 gives the information about number of post and illustrations for some entities of our dataset.

B. Outcome:

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

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

TABLE I DETAILED INFORMATION OF COLLECTED DATA AND GENERATED CLUSTERS

#Posts	#Comments		#Retrieved Clusters	
	BatchSTS	***Incremental***	***BatchSTS***	***Incremental***
20	407	420	307	307
20	346	355	248	253
08	216	230	93	103

"Fig. 2" shows that the number of clusters generated using BatchSTS method goes on falling as radius threshold *k* increases and vice-versa. Thus it can be said that if *k* is set to a better value then time required for clustering is less. But the better *k* value is more appropriate in the case of opaque data. In case of thin data, *k* should be little to protect cluster resemblance, but this increases time complication of clustering process.

"Fig. 3(a)" shows the number of clusters generated by BatchSTS method depending upon number of input illustrations, resemblance between those illustrations & radius threshold value *k*.

"Fig. 3(b)" shows the number of clusters generated by Incremental method. In 1st case, numbers of illustrations are increased as compared to 1st case in BatchSTS, due to the

adding of new comments. But the number of clusters has remained similar, as all the recently added illustrations were similar to the preceding clusters. So only the obtainable clusters are updated. In 2nd & 3rd cases, the number of clusters has increased because of two reasons as: not all of the newly added illustrations are alike to the obtainable clusters or recently inwards illustration is alike with some cluster but cannot be added into it if its adding 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 illustrations, their resemblance with available BatchSTS clusters & radius threshold value *k*.

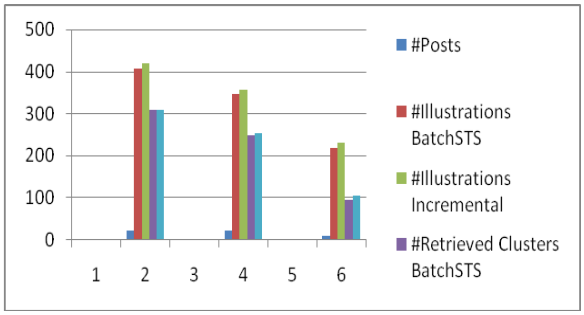

Fig.3: Variation in number of clusters according to number of illustartions (a)BatchSTS (b)Incremental

V. CONCLUSION

Thus, it seems that, for real time graphic stream illustration problem on social networking sites, incremental clustering technique proves to be valuable. Incremental clustering process considers newly received illustrations, so the clusters are obtained in incremental manner. These clusters are then summarized so that users can get an summary of all illustrations easily and speedily as a substitute of evaluation the set of comments of each job. Thus user obtains updated recapped at any moment. Accessible clustering methods which are presented here mainly spotlight on maintaining cluster quality hence they cannot provide real time updated recapped of illustrations. But the technique in this manuscript satisfy this purpose of obtaining incremental recapped of comment sets on various posts, on social network sites and hence reduces the user's hard work. In outlook, use of speech semantics can be considered to give more suitable recapped results.

VI. REFERENCES

[1]. Cheng-Ying Liu, Chi-Yao Tseng, Ming-Syan Chen, " IncreSTS: Towards Real-Time Incremental Short Text Summarization on Comment Streams from Social Network Services " *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, No. 11, November 2015.

[2]. J. A. Hartigan and M. A. WonG, "A K-means clustering algorithm", J.Roy. *Statist. Soc. Series C (Appl. Statist.),* vol. 28, no. 1, PP. 100108, 1987.

[3]. K.D.Rosa, R.Shah, A. Gershman, R. Frederking, "Topical clustering of tweets", *in Proc.ACM SIGIRs Social Web Search Mining*, 2011.

[4]. D. T. Pham, S. S. Dimov, and C. D. Nguyen, " An incremental k-means algorithm", *Proc. Institution Mech. Eng., C, J. Mech. Sci*, vol. 218, no. 7, 2004.

[5]. N. Sahoo, J. Callan, G. Duncan, R. Krishnan," Incremental hierarchical clustering of text documents", *in Proc. 15th ACM Int Conf. Inf. Knowl. Manag*, pp. 357366, 2006.

[6]. E.Khabiri, J.Caverlee," Summarizing User Contributed Comments", *Proc. of 5th International AAAI Conference on Weblogs and Social Media* 2011."

[7]. J.-Y. Weng, C.-L. Yang, B.-N. Chen, Y.-K. Wang and S.-D. Lin" IMASS: An intelligent microblog analysis and summarization system", *in Proc. ACL/HLT Syst. Demonstrations*, 2011.

[8]. M. F. Porter," An algorithm for suffix stripping", *Program*, vol. 14, pp. 130137, 1980.

[9]. Pooja S. Choudhari, Prof S. S. Nandgaonkar," Short Text Summarization Of Comment Streams On Social Network Sites", *International Journal of Computer Systems*, Volume 03, Issue 07, July, 2016.