**Algorithm 1: To convert messages into appropriate topics**

1. **Input :** JsonParsedMessages, WindowSize

2. **Output :** TopicSet

3.  **Initialization :** TopicSet=0,

4. **for** each message i **do**

5. topic[i]=None

6. **end for**

7. **for** each message index i **do**

8. **if** i>0 and author of message[i] is same as before **then**

9. Merge {topic[i], topic[i-1]}

10. **else**

11. **if** message[i] is a replyObject **then**

12. Merge {topic[i], topic[i-1]}

13. **else**

14. Start a new topic

15. **end if**

16. Add the new topic to the window of size WindowSize

17. **end if**

18. **end for**

19. **for** each topic in TopicSet **do**

20. **if** number of messages in topic is <= 2 **then**

21. Absorb the smaller topic in larger one

22. **end if**

23. **end for**

**Algorithm 2: To Predict if current message is a reply object or not**

1. **Input :** message, thresholdForSimilarity

2. **Output :** Topic to which the current message is

a reply, along with appropriate reason**.**

3. **Initialization :** replyStarters = ['ok', 'ok.', 'k', 'k.' 'mine', 'his', 'hers', 'theirs', 'ours']

3. **if** any topic is present in window **then**

4. Calculate SimilarityDistance of current topic with other topics in the window

5. Tokenize the message

6. Find POS tag of the tokens

7. **if** length of tokens <= 1

8.  **or** POS tag is starting with a universal reply tag

9. **or** message starts with a replyStarter

10. **or** message has @UserID in it

11. **or** size of last topic == 1

12. **or** similarityDistance > thresholdForSimilarity **then**

13. Classify it as a reply object with appropriate topic and reasoning

14. **else**

15. Classify it as not a reply object

16. **end if**

17. **end if**

**Algorithm 3: To find similarity Distance of messages in the window**

1. **Input :** message, windowTopics

2. **Output :** Similarity Score

3. **Initialization :** slidingWindowSize = 10, scoresArray = []

4. Define decay function as decay = 0.993\*(difference of IDs of messages)

5.  **for** each message in each topic of windowTopics **do**

6. Tokenize the message

7. Convert tokens to vector

7. **for** each vector of tokens according to slidingWindowSize **do**

8. Calculate sum of absolute values to give Euclidean distance

9. Calculate cosine score

10. Store the best pair of both values

11. **end for**

12. Add the pair of values to scoresArray

13. **end for**

14. Sort on the basis of scoresArray[‘eucideanDistance’]

15. Take out only top 5% of the scores from the array

16. Sort on the basis of scoresArray[‘cosineScore’]

17. Classify similarity score as the scores from first index of the sorted array