1. Go to spark installation folder and then create a folder for your code. I compiled and ran my code using SBT and vi editor. Create .sbt file to tell sbt which version of scala to execute and its library dependencies.
   Create a folder structure source/main/scala. Place your scala code here.

   To compile give :sbt package
   To Run: /Users/Alya/workspace/spark-1.3.0-bin-hadoop2.4/bin/spark-submit --class "LDAVer2" --master local[*] target/scala-2.10/simple-project_2.10-1.0.jar
Jar file is in target/scala-2.10

I ran my code on local machine. First I tried on one part file and it worked fine.But when I try to work on entire corpus it took almost 8 hours ending up out of heap memory.

I tried working by dividing 5 GB in to 1 GB each using
cat twitter-processed/part-1* > tweet1.txt
cat twitter-processed/part-2* >tweet2.txt
cat twitter-processed/part-3* >tweet2.txt
cat twitter-processed/part-4* >tweet3.txt
cat twitter-processed/part-5* >tweet5.txt

But I still faced same problem.

In this file I am submitting results on 10 and 26 part files. I know that this is really small sample, but I think I made mistake by choosing to run on local machine(which has limited RAM and slow in processing) rather than z cluster with multiple slaves.
Although I tried many options for memory management in code i don't see much improvement in the performance may be because of limited processing power of my laptop.(Can see the memory management options I used in my code in spark configuring(
eg: In LDAVer2.scala, I used

```
val conf = new SparkConf()
                .setAppName("LDA Ver2")
                .setMaster("local[*]")
                .set("spark.executor.memory", "2g")
                .set("spark.shuffle.memoryFraction","0.001")
                .set("spark.rdd.compress","true"))
```

As mentioned above I have 2 samples data,I sampled them using
Sample1: cat twitter-processed/part-0115* > /Users/Alya/Desktop/Final/twit10.txt (part files from part-01150 to part-01159)
Sample2 : Concatenated first 26 files in to single document and ran LDA on it.(part-00000 to part-00025)

Note: I am giving unique words on different sample because I was able to calculate only unique words for large files on my local machine. I was able to run LDA and report top words only on 2 small sample data sets. When I try to run LDA on large data set I got Out of Heap memory exception.

So the following are results I got

1. Number of Unique words for 10 files = 24785
   Number of unique words for 26 files = 48741
   Number of unique words for 50 files = 149782
   Number of unique words for first 3000 files = 2077454

2. I ran LDA on 10 part files with k=10 topics.
   The following are the top words from 10 topics.
   Top word in topic 0 is rt
   Top word in Topic 1 is rt
   Top word in Topic 2 is rt
   Top word in Topic 3 is rt
   Top word in Topic 4 is rt
   Top word in Topic 5 is rt
   Top word in Topic 6 is rt
   Top word in Topic 7 is rt
   Top word in Topic 8 is rt
   Top word in Topic 9 is rt

I see rt(stop word in twitter) is dominated by the important data. I cannot say much about the data as i see rt as top listed word.

LDA on 26 part files with k=10 topics
   Top word in topic 0 is rt
   Top word in Topic 1 is rt
   Top word in Topic 2 is rt
   Top word in Topic 3 is rt
   Top word in Topic 4 is rt
   Top word in Topic 5 is rt
   Top word in Topic 6 is rt
   Top word in Topic 7 is rt
   Top word in Topic 8 is rt
   Top word in Topic 9 is rt

Same as above, I only see rt where I can not infer any thing.

3. Since I can not infer much from data because of stop words, I need to remove them from the data set to categorize data in to top topics.
I used 108 stopwords. The following are the list of stop words used.

("rt","a","an","the","able","about","after","all","almost",
"also","am","and","any","are","as","at","be","because","been","but","by",
"can","cannot","could","dear","did","do","does","don't","either","else","ever","every","for","from",
"get","got","had","has","have","he","her","him","his","how","however","i","if","in","into","is",
"it","like","may","me","might","most",
"must","my","no","nor","not","of","on","only","or","other","our",
"own","rather","said","say","she","should","since","so","some","than","that","the","their","them","
then","there","these", "they","this","to","too","was","us","was", "we","were","what","when",
"where","which","while","who","whom","why","will","with","would","yet","you", "your")

4. The code for removing the stop words from input and performing LDA is in RemoveStopWords.scala (inside LDANoStopWords/src/main/scala)
After removing stop words, Unique words for 10 part files data set are 24680.
I see 105 stop words were removed from the data.
For 10 topics the following are the top words.(For 10 part files)

Top word in topic 0 is shots
Top word in Topic 1 is shots
Top word in Topic 2 is shots
Top word in Topic 3 is drunk
Top word in Topic 4 is broken
Top word in Topic 5 is shots
Top word in Topic 6 is shots
Top word in Topic 7 is drunk
Top word in Topic 8 is drunk
Top word in Topic 9 is drunk

From this I can probably infer that the corpus/data is about drinking alcohol, shots of alcohol. This might not be 100% true but from words which are used mostly, there are high chances that these are about tweets related to alcohol. And top topic wordss are shots, drunk and broken

For 26 part files after removing stop words, I see unique words as 48636 words. I see here also 105 stop words were removed from the data.
Top words for top 10 topics are

Top word in topic 0 is shots
Top word in Topic 1 is shots
Top word in Topic 2 is drunk
Top word in Topic 3 is drunk
Top word in Topic 4 is .
Top word in Topic 5 is drunk
Top word in Topic 6 is drunk

Top word in Topic 7 is drunk
Top word in Topic 8 is drunk
Top word in Topic 9 is drunk

I can still infer that these documents might belong to drinking and alcohol. Also top topic words are drunk and shots.

5. Now that we have trained LDA with sufficient data, our trained model is ready. If a new tweet arrives then we can remove the stop words from it and calculate probability of each word. Then arrange the words in the order of decreasing order of probability. If the top words with higher probability in testing model coincide or look similar to training model, then you can classify the tweet in to same category. Otherwise the tweet is classified in different one.
To see training and testing documents are similar, I take the top words from each topic in LDA. Number of topics are chosen as k value(Here 10 topics) and compare it against the word with top probabilities.
Thus we can classify/cluster new tweet by saying which topic does it belong to.