In [1]:

```
from pyspark import SparkConf, SparkContext, SQLContext
from pyspark.sql import SparkSession
from pyspark.ml.feature import Word2Vec, CountVectorizer
from pyspark.ml.clustering import LDA, LDAModel
from pyspark.sql.functions import col, udf
from pyspark.sql.types import IntegerType, ArrayType, StringType
import pylab as pl
```

In [2]:

```
def to_word(termIndices):
    words = []
    for termID in termIndices:
        words.append(vocab_broadcast.value[termID])
    return words
```

In [3]:

```
# Load document dataframe (provided by the TA)
PATH = "gs://6893_course_data/twitter_data/stream_data.csv"
spark = SparkSession.builder.appName("LDA").getOrCreate()
spark_df = spark.read.csv(PATH)

spark_df.show()
```

```
+--------------------+
|                 _c0|
+--------------------+
|I absolutely ADOR...|
|Java Vs Python Fo...|
|voulu un grec pui...|
|Pareil Il pris de...|
|Music Academy Blo...|
|Tarps, tents, and...|
|voulu un grec pui...|
|We drive efficien...|
|Check out my Gig ...|
|Hey, nice bones y...|
|lembro como sofri...|
|WHO WITH A DEEP T...|
|@Tina69911364 @As...|
|alguem cria um ap...|
|@Neptvn08 Comment...|
|une dinguerie de ...|
|Y a une grosse mo...|
|Je te cache pas q...|
|@JAPANFESS setauk...|
|Femme recherchant...|
+--------------------+
only showing top 20 rows
```

In [4]:

```
# dataframe preprocessing
from pyspark.sql.functions import col, split
spark_df = spark_df.withColumnRenamed('_c0', 'words')
spark_df = spark_df.withColumn("input", split(col("words"),"\s+"))
spark_df.show()
```

```
+--------------------+--------------------+
|               words|               input|
+--------------------+--------------------+
|I absolutely ADOR...|[I, absolutely, A...|
|Java Vs Python Fo...|[Java, Vs, Python...|
|voulu un grec pui...|[voulu, un, grec,...|
|Pareil Il pris de...|[Pareil, Il, pris...|
|Music Academy Blo...|[Music, Academy, ...|
|Tarps, tents, and...|[Tarps,, tents,, ...|
|voulu un grec pui...|[voulu, un, grec,...|
|We drive efficien...|[We, drive, effic...|
|Check out my Gig ...|[Check, out, my, ...|
|Hey, nice bones y...|[Hey,, nice, bone...|
|lembro como sofri...|[lembro, como, so...|
|WHO WITH A DEEP T...|[WHO, WITH, A, DE...|
|@Tina69911364 @As...|[@Tina69911364, @...|
|alguem cria um ap...|[alguem, cria, um...|
|@Neptvn08 Comment...|[@Neptvn08, Comme...|
|une dinguerie de ...|[une, dinguerie, ...|
|Y a une grosse mo...|[Y, a, une, gross...|
|Je te cache pas q...|[Je, te, cache, p...|
|@JAPANFESS setauk...|[@JAPANFESS, seta...|
|Femme recherchant...|[Femme, rechercha...|
+--------------------+--------------------+
only showing top 20 rows
```

In [5]:

```
# CountVectorizer
cv = CountVectorizer(inputCol="input", outputCol="features")
model = cv.fit(spark_df)
cvResult = model.transform(spark_df)
cvResult.show(5)
```

```
+--------------------+--------------------+--------------------+
|               words|               input|            features|
+--------------------+--------------------+--------------------+
|I absolutely ADOR...|[I, absolutely, A...|(4475,[0,9,12,62,...|
|Java Vs Python Fo...|[Java, Vs, Python...|(4475,[241,398,71...|
|voulu un grec pui...|[voulu, un, grec,...|(4475,[8,14,15,55...|
|Pareil Il pris de...|[Pareil, Il, pris...|(4475,[2,13,15,21...|
|Music Academy Blo...|[Music, Academy, ...|(4475,[0,3,4,30,1...|
+--------------------+--------------------+--------------------+
only showing top 5 rows
```

In [6]:

```
# train LDA model, cluster the documents into 10 topics
ldaModel = LDA(featuresCol="features").setK(10).fit(cvResult)
```

In [7]:

```
transformed = ldaModel.transform(cvResult).select("topicDistribution")
#show the weight of every topic Distribution
transformed.show(truncate=False)
```

```
+------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------+
|topicDistribution
|
+------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------+
|[0.005186164216874515,0.003960160476691635,0.003956312707033753,0.003934193713513273,0.003934219359069845,0.003934507045586802,0.962
9163772715503,0.004071676643242336,0.0039378703542221295,0.004168518212215463]   |
|[0.007799073061056249,0.9418017624877411,0.005949954723848018,0.005916704936338126,0.005916747034640611,0.005917171856976650,0.008383
900180722291,0.006123461481739278,0.005922236727356218,0.006268987509581555]     |
|[0.9436470323598692,0.005955713934380984 5,0.005949909186700705,0.005916661523791629,0.005916698285244 34,0.005917129790119107,0.00838
2368015919312,0.006123399724134416,0.005922191400707887,0.006268895779132182]    |
|[0.9536550689576917,0.004897957061174042,0.004893177872296634 5,0.004865831164086 37,0.004865866670514 05,0.004866217930864015,0.006894
118395051148,0.005035861912325537,0.004870379093801928,0.005155520942194619]     |
|[0.006057495012731839 4,0.004624384862931298,0.004619891300247162,0.004594043698171925 6,0.00459410613815166,0.004594421039009235,0.95
66949820658037,0.004754626289906503,0.0045983438445779,0.004867705748468703]     |
|[0.008404860455611665,0.006417818674753064,0.006411558651371602,0.006375745815758958,0.006375775518467542,0.006376235276310497,0.366
04517638977097,0.006598539563966586,0.006381690980193671,0.5806125986737956]     |
|[0.9436470323598692,0.005955713934380984 5,0.005949909186700705,0.005916661523791629,0.005916698285244 34,0.005917129790119107,0.00838
2368015919312,0.006123399724134416,0.005922191400707887,0.006268895779132182]    |
|[0.007801270400857256,0.005955939362672 46,0.005950136191427075,0.005916895956584153,0.7620980170657207,0.005917359032440619,0.188044
84003716423,0.006123742248290 6334,0.005922403279368215,0.006269396425474574]      |
|[0.006057977677200088,0.004624389387720662 5,0.004619878618766 66,0.004594052238250576,0.004594230867509453,0.004594433119314479 5,0.95
66942091602015,0.004754686302647543,0.004598367994717345,0.004867774633671651]   |
|[0.007281311095216425,0.005556029560773472,0.005550633049020 38,0.005519588670424111,0.005519645752300187,0.005520022994663279,0.9479
67104804293,0.005712525314545051,0.0055247218614868695,0.005848416897277269]      |
|[0.005735702481246 09,0.004379419980987283 4,0.004375144655380202 4,0.004350697309650789,0.004350728770987631,0.004351045407472334,0.95
89899681443459,0.004502794310345266,0.00435476567208755,0.004609733267496938]    |
|[0.004733038936727789,0.003614172954162737 2,0.003610654342886255 4,0.003590475477526231,0.003590504026523226 3,0.003590759097359575,0.
9661563207066061,0.003715944956293797,0.0035938294294121167,0.0038043000725021545]|
|[0.008404503425467393,0.006417785925041724,0.006411527617476171,0.006375703612792339,0.006375747350987522,0.006376206899185855,0.009
03998868325186,0.006598503995542425,0.0063816655993910055,0.9376183668908636]     |
|[0.007803199636703919,0.005955906308488051,0.005950068058683891,0.005916811065958371,0.005916875133244223,0.005917293939035641 5,0.51
21368840098761,0.43821143312845695,0.005922340426761541,0.006269188292791376]    |
|[0.9474273442432625,0.005555882602923386,0.005550473711243357,0.005519464393656279,0.005519500764727213,0.005519895459695997,0.00782
2166560598687,0.005712410802207438,0.005524682614590265,0.00584817884709508]     |
|[0.006059656566684649,0.004624167462850101,0.004619662627856 73,0.004593843735915511,0.004593874880659 69,0.004594206496118 98,0.956694
7530799059,0.004754372622805529,0.004598136797790713,0.004867325729412239]       |
|[0.9436366740589319,0.005955999721701497,0.005950187627745641,0.005916929227470806 5,0.005916953467225662,0.005917449632176018,0.0083
90136535726239,0.006123814679523933,0.0059224539348817235,0.006269401114616579]   |
|[0.9474250026351184,0.005555943097069845,0.005550516003196437,0.0055194991182971 4,0.005519530188958094,0.005519917021080304,0.007824
410227391599,0.005712397056635345,0.005524639594754359,0.005848145057498606]     |
|[0.009111281561980 15,0.006957492920547816,0.006950724315800845,0.006911870994411143,0.006911904782800206,0.006912413514777064,0.0097
96179662681438,0.007153390930546373,0.006918321696266507,0.9323764196201885]     |
|[0.9436443127016174,0.005955790992779921 5,0.005949979531867567 6,0.005916728337001588 5,0.005916773076453810 5,0.005917195452316469,0.0
08384455766608265,0.006123486927938401,0.005922261629643798,0.006269015583772888] |
+------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------+
only showing top 20 rows
```

In [8]:

```
#The higher ll is, the lower lp is, the better model is.
ll = ldaModel.logLikelihood(cvResult)
lp = ldaModel.logPerplexity(cvResult)
print("ll: ", ll)
print("lp: ", lp)
```

```
ll:  -123598.70801748236
lp:  11.011020758795757
```

In [9]:

```
# Output topics. Each is a distribution over words (matching word count vectors)
print("Learned topics (as distributions over vocab of " + str(ldaModel.vocabSize())+ " words):")
topics = ldaModel.topicsMatrix()
print(topics)
```

```
Learned topics (as distributions over vocab of 4475 words):
DenseMatrix([[24.1896685 ,  1.18594903,  0.56720227, ...,  0.83336772,
               1.54761506,  0.68031076],
             [ 0.59859061,  1.24843117,  0.69348586, ...,  0.57424243,
               0.56883441, 12.28921633],
             [76.35448255,  0.63714835,  1.31674193, ...,  4.21452495,
               0.54687653,  0.56760941],
             ...,
             [ 0.76212978,  0.63671608,  0.54815412, ...,  0.74086032,
               0.6261614 ,  0.65348755],
             [ 0.54197475,  0.68193242,  0.61783876, ...,  0.6347725 ,
               0.59419805,  0.87816495],
             [ 0.57097721,  0.64350979,  0.7138692 , ...,  0.59503576,
               0.60042012,  0.51342917]])
```

In [ ]: