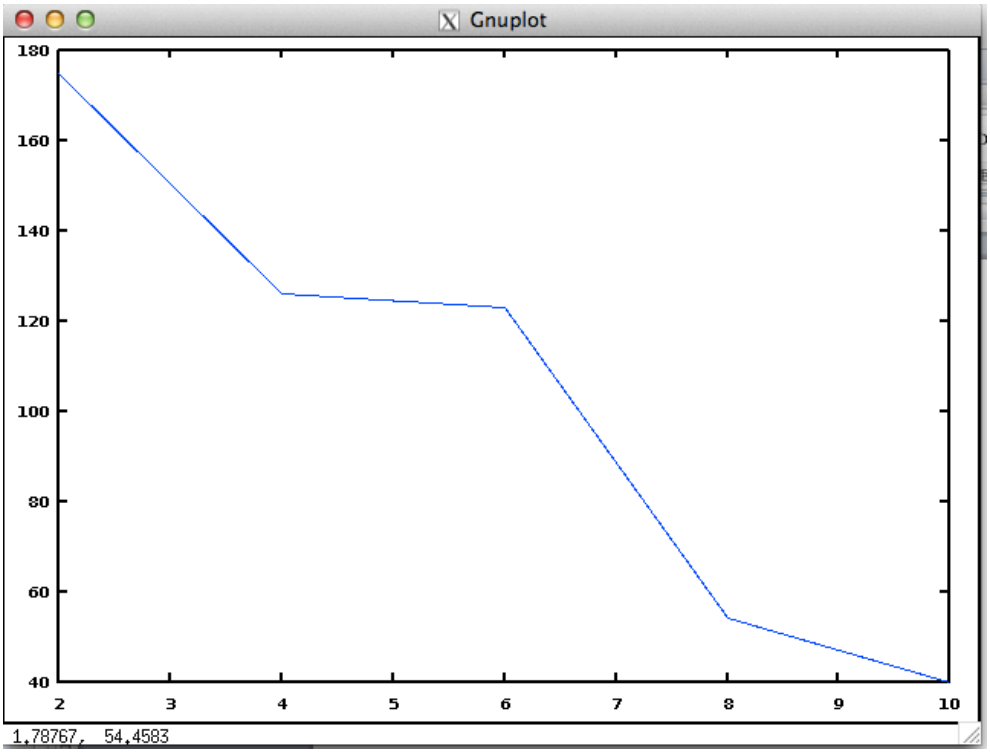
1,

The unit of time is second.

Reducers	2	4	6	8	10
Time(second)	175	126	123	54	40



The wall time is not linear with respect to the number of reduces.

When the reducers changed from 2 to 4. The time changed from 175 to 126, which is not reduced by 50%.

When the reducers changed from 4 to 8. The time changed from 126 to 54, which is reduced more than 50%

2,

reducers:10

Time: 7 minutes

instance type: m1.small

location: N.Virginia

3,

When the result of training is not that large and we can load to the memory.

We can just load the dict to memory and for each test data. We can just extract the prior and likelihood from the dict and posterior can be calculated.

When the dict is too large to be loaded to memory. We can use MapReduce. We can streaming both the test data and the dict. But with some tag on the test data to separate the test data and the dict. So that in the reduce step, we can extract the prior and likelihood, and we can know which is the test data. In this way we can calculate the posterior.

4,

Yes. It will adversely impact the performance of the MapReduce job. Because when there are some words occur much more in the corpus, these words will be hash to the same reducer. Then this reducer will have a heavy job, which other reducers have much lighter job. Then, all reducers which have light job have to wait for the heavy ones to finish.

5,

It's better to merge the documents before move to HDFS. Because the every file. There is some overhead for HDFS to manage it. And the maximum size of block on HDFS is 64M. When the size of file is less thatn 64M, it still going to use 64M. IT's a waste to put just 5kb file in a 64M block. And also It's slow to read multiple small files than one bigger one.

6,

(1) e
(2) f
(3) i
(4) c
(5) h
(6) g
(7) j
(8) d
(9) a
(10) b