ReadMe for Code:

We used the following external jar to convert the json and read the hashtags.

```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.7</version>
</dependency>
```

The classes Tweet, Entities and Hashtags are defined in the respective files.

For the Majority Algorithm,

Build command: javac MajorityAlgorithm.java

Run command: java MajorityAlgorithm.java

For the countMin sketch

Build command: javac CountMin.java

Run command: java CountMin.java

The output for the program should be available in CountMin_Output and MajorityAlgorithm_Output text files.

A comparison of Time and Space Complexities

Space Complexity

The Majority Algorithm requires an array of size k(500) to implement while CountMin uses a Two Dimensional array of size height x width(20x2000) and minHeap which contains nodes whose frequency greater than m/k. Hence the majority algorithm consumes less space than Count Min Sketch.

Time Complexity

The time complexity of the majority algorithm would be O(mk). Because in the worst case when an element occurs and is not in the array it decrements the count of every element in k, and since there are m elements => O(mk)

The time complexity of CountMinSketch would be as follows, Inserting into countMin data structure takes O(t) runtime. Because insertion into an array is O(1) and we do it for t(height) hash functions. All operations on Minheap is done by O(log minheapSize) at max. Hence all operations on countMinSketch can be performed at O(mt) at max.

Comparing Majority Algorithm with CountMinSketch, t(No of hash functions) is always lesser than k. Hence CountMinSketch performs better than Majority Algorithm in case of time complexity.