Behavior:

1. Absence of some words like cuboulder and iphone6 in the dictionary"

2. There are some words in the dictionary that influnces the result like ue, isa, inn"

3. There are some words that comprises of two or more words like forthe-> for the, superbowl->super bowl"

4. Sometimes the algorithm, in the process to match the maximum length words, in the tail leaves letters that couldn't be matched, like vsat l, ol l"

5. Only two one letter words have been used. they are u and i. Other one letter words can be removed."

Improvements:

Changing the lexicon:

1. After analyzing the behavior of the algorithm it was seen that only two one letter words were used in the hashtags. So all the other one letter words were removed
2. Only a few frequently used two letter words in the English language were retained in the dictionary. The others were removed.
3. Only a few three letter words used most frequently in the English language were used. Others were removed. The reference was : http://scottbryce.com/cryptograms/stats.htm
4. Some compound preposition words like tobe, ofthe were deleted.

5. Some words like ipad, mentalist were added that were not in the dictionary.

Modifying the strategy:

1. First the improvement was made in this way: the length of the longest word in the dictionary was noted as maxlength.. Then the patterns from the hashtags were made such that each token could not exceed the length I where I started from 1 to maxlength. This approach gave the WER value of 0.19375. By checking the outputs, it was seen that because the length of the tokens in each pattern were limited, most pharses like climate change were shown as clima tech. so the idea was then to partition the hashtags into two places and then keep partitioning them and keep finding the longest matching patterns in each of those two partitions.
2. First the hashtag were partitioned in such a way that it gets all the possible patterns for the given hashtag. To do this, we initialized an index as the start of the string. Then we partitioned the hashtag from start to the index and from the index to the last place of the string. Thus we have two partitions. Now the maxmatch algorithm was applied twice for the two partitions and all the matches were appended to the possible matches list. After completion of an iteration, the value of the index was incremented once until it reached the length of the index. After finding out all the patterns, the pattern that had the number (#number of tokens -#number of words matched in the dictionary) as the lowest was taken.
3. The next improvement was to what to do when there is a tie between two matches with having the same number of (#number of tokens -#number of words matched in the dictionary). To handle these, it was first checked for each match whether last letter of one token can be prepended to the next token and still can be eligible

for a word (Colbert late show), (Colbert lates how). Here the s from the secnd token can be added with how to make the word show which is also a valid word. If a new match comes in with tokens like this and also having the number (#number of tokens - #number of words matched in the dictionary) equal to the previous best match then that match was taken.

4. The next step was to check if the wordcount is the minimum when there is a tie. This was to make sure we always take the longest token-words in a match. This was the modification of the greedy strategy. By doing these improvements, the WER was computer as 0.03.

WER values:

1. Before the improvements: 0.53125
2. After the improvements 1 and 2: 0.0875
3. After improvements 3 and 4: 0.03125

WER Values on the test set that was given on 9.17.2015(today):

1. Before any improvements: 0.3833
2. After improvement 1 and 2: 0.366
3. After improvement 3 and 4: 0.21

I compiled the answers for the test set myself which might be different from it actually is. The code was prepared and the behavior was analyzed using the test hash tags that were provided earlier and which was the test case for last year.