CS771A: Assignment 2

Vivek Agrawal(201133)
Shubhum Kumar (200967)
Gavish Garg (200385)
Amanjit Singh (200107)
Vibhor Diwakar (201107)
Praveen Kumar Singh (200719)

Question 1:

Results:

1. Average Query Count = 4.0
2. Average Train Time = 9s
3. Win rate = 1.0

Data Structure used for node:

- We kept a list (*all_words*) of all words in the dictionary given
- A list (*my_words_idx*) which stores the index of all words which are possible from that node. These are the indices of the words in the *all_words* list
- We also keep *history* at each node which stores information regarding all the queries made before reaching the current node from the root.

Splitting criteria for each node:

The query word at each node is beforehand stored in the trained tree which we get. This query word is passed to the Merlin which after checking with the secret word reveals some characters common to both the strings. This response is then used to decide to go in which children node from the current node. We have created children nodes for each possible response from that node. All the words which have the same respone when queried by the query word go to the corresponding children node. This is thus the splitting criteria for words at the node.

Algorithm used to decide the Query word:

As discussed above, we store the index of all possible words reachable from that node in the list *my_words_idx*.
Now, to decide the query word, we consider all the words with indices in the *my_words_idx* list but have not been made query earlier. To check this, we remove all the words in the history from our consideration. We save these indices in the list *available_query_idxs*.
Consider any query word **query** from this list.
We apply reveal function for all the words at the node (stored in *my_words_idx*). The returned value (*mask*) is the incompletely filled string (response). We use these as key to create a dictionary of lists. List for a entry in dictionary stores the indices of words which gave the same response on applying the query. This thus divides the words at a node into groups which are then created as children nodes. We calculate the entropy change for this division and append it to a list.
The query word which gave the the most entropy change is stored as the query word for the trained model at this node. History is updated (added) with the query word for this node.

The query index and split dictionary is returned by the *process_node* function.

Note: We selected the query word for a node from the subset of words present at that node and not from all the words in the dictionary is because when we implemented this way, we got worse results. The average query count increased to 4.3 also the average train time did not decrease.

Criteria to make a node as leaf:

If any of the below two conditions hold we stop expanding the decision tree and make that node as leaf node:-

1. The count of words at the node is equal to 1.
2. Depth of the node is 15 (maximum number of queries which can be made).

Note:

To decrease the average query count, we implemented a lookahead of 1 (which means we decided the best query at a node by seeing the change in entropy for 2 levels of the tree). But this did not give better results, in fact increased the average query count to 4.5. So, we removed the lookahead and implemented without it.