CS771 ASSIGNMENT 2

Team : Veg_Korma

Ashutosh Agrawal (210219)

Hitesh Anand (200449)

Pratham Jain (200712)

Priya Gole (200727)

Shambhavi Sabharwal (200914)

Tanush Kumar (201043)

1 Problem - 1

At each node, we have a set of words representing the collection of possible answers.

We perform the following steps in order to choose the query to be made at each step :(in the process_node() function)

- Let L be the set of words at any node (represented by my_words_idx). For each possible word guess (let's call it 'query') ∈ L, we find possible responses from Melbot by calling the reveal() function on other words (let's call it 'word') in the set L. Formally, the reveal(word, query) function returns a string mask such that $mask_i = word_i$ for $word_i$==$query_i$, and $mask_i$ = '_' otherwise. In other words, this function returns Melbot's response when the target string is *word* and our guess is *query*.

- This creates a partition of L into n (say) sets $\{Y_k\}_{k=1}^n$. For each possible word guess w ∈ L, we get a partition. Words belonging to a partition give the same string mask with the chosen word w. Therefore, there are O(|L|) partitions of L.

- Intuitively, a good guess will result into $\{Y_k\}_{k=1}^n$ being sets of strings with small cardinality. We assign a score of $\sum_{k=1}^n |Y_k|^2$ to each of the partition created. This score acts as our decision criteria while splitting a node.

- We select the word w ∈ L, that leads to the minimum score among all possible guesses. We create a partition based on this w, and perform steps 1-4 recursively. The intuition behind this is the same as stated in the last point. Such a selection will result into a more balanced split in the current node.

- The above process terminates when a particular leaf has only 1 word left. In that case, process_leaf() is called and we return that word as our final guess.

1.1 Bookkeeping for Reducing Training Time

Previous research[1] shows that creating the mask for each pair (word, query) by calling the reveal() function is the bottleneck in the training process. Hence, we try to reduce the training time by bookkeeping the masks for each pair (word, query). We do this by storing the value of mask after it is encountered for the first time. As the table in next section shows, this reduced the training time by almost half. However, there was a tradeoff in terms of avg queries and model size. For this bookkeeping, a 2D array was created which was passed along the children of the current node (while calling the fit() function recursively). Hence, this leads to increase in the model size. Our final model doesn't perform any bookkeeping since we decided to go for lesser number of avg queries and model size with a slightly more training time.

1.2 Results

Condition	Avg Queries	Training Time	Model Size	Accuracy
Without Bookkeeping	5.18	4.87s	1419927	1.0
With Bookkeeping	4.04	8.13s	1136610	1.0

2 References

1 Mathematical optimization over Wordle decision trees