CS771

The Neural Nuts

April 2023

Problem 1

Solution : We can solve the Melbot challenge using a decision tree to guess the word in the least number of queries required.

Structure of Decision Tree

Node: Each node of the tree contains the following:-

- Depth: Stores the depth of the tree
- Parent: A link to the parent node
- My Words:- Stores the indexes of the words that have reached the node
- Children:- A list of all children of the node
- Query index: It stores the index of the query at this node
- Is Leaf: Specifies whether a node is a leaf or not

Tree: The class Tree has the following attributes:-

- Root: Stores the root of the tree
- Words: Stores the entire dictionary
- Minimum Leaf Size: Stores the minimum size of the leaf
- Max Depth: Stores the maximum depth allowed/number of queries allowed

Concepts Used

We have used Information Gain (IG) index to determine the best query at the node present. Information Gain (IG) is the difference in entropy between the children node and the parent node.

$$IG = \Delta(entropy)$$

Now, for a given query word, we have classified the words based on the mask it forms with the query word. Hence, the IG of a node will be:-

$$IG = \sum_{t \in allmasks} \frac{n_t}{N}(-log_2(\frac{n_t}{N})$$

Here, n_t is the number of words belonging to mask t (considering mask as the splitting criteria), and N is the total number of words at the Node.

Splitting Criterion

Root Node: At the root node, we have divided the words based on their lengths and words with the same length get classified in the same class.
We send an empty string ("") as the query, so Melbot returns an empty mask equal to the length of the word. Using this, we can easily predict the length of the word and hence go to the corresponding child node.

Strategy: At each node, we send a query and get a mask associated with it. Each child represents a possible mask that can be obtained from the query, and we proceed to the child with the same mask as given by the reveal function.

During training time, at each node, we send every word present at that node as a query and calculate the Information Gain or reduction in entropy and select the one which results in the maximum Information Gain.

Stopping Criteria: We make a node leaf if it contains only one word. We will stop as soon as we hit a leaf, as that word will be the required answer. Also, the maximum depth of the tree is set to be 15 as we are allowed to ask 15 queries at max, and if a node's depth reaches 15, we make it a leaf.

Hyperparameter: As various words will result in similar masks and will have almost equal information gain, so instead of iterating over every word, we can iterate over a smaller subset of the words chosen at random, which will mimic the distribution of all the words. This will drastically reduce the training time while the performance will not be degraded drastically (as depicted below).

We introduce a hyper-parameter λ which is:-

$$\lambda = \alpha N$$

where N=total number of words in the dictionary

Let n be the total number of words in the current node. Let

$$k = max(n, \lambda)$$

In our model, we are iterating over k randomly chosen words out of the total words in the node to find the best query to maximise the Information Gain. We ran it over different values of α:

Value of α	Training Time	Size	No. of Queries
0.002	0.5413	603225.6	4.2312
0.005	0.9566	599004.0	4.1372
0.01	1.1557	597466.6	4.0974
0.1	6.8299	594486.2	4.0051
0.3	8.4114	594235.0	3.9990
0.5	8.6856	594235.0	3.9990
0.8	8.4136	594235.0	3.9990
1	8.4012	594235.0	3.9990

The win rate (i.e. the number of times the word was guessed correctly within 15 queries) for all these models was 1.

Problem 2

Solution : Implemented in **Submit.py**