CS771A Assignment 2

Adit Jain
Junior Undergraduate
Dept. of Electrical Engineering
Roll no.: 200038
aditj20@iitk.ac.in

Akshit Verma
Junior Undergraduate
Dept. of Electrical Engineering
Roll no.: 200091
akshitv20@iitk.ac.in

Ahmad Nabeel
Junior Undergraduate
Dept. of Material Science Engineering
Roll no.: 200063
ahmad20@iitk.ac.in

Abstract

This document is the submission of our group, "No Brainers" for Assignment 2. We have answered Part 1 with all the relevant level of detail required, providing mathematical proofs wherever required.

1 Solution to Question 1

1.1 Introduction

The first step in designing a decision tree algorithm for this problem is to determine the splitting criterion for each internal node. At each internal node, Melbo needs to select a query word that can effectively differentiate between the remaining candidate words in the dictionary. To determine the best query word, we can use the information gain criterion.

Information gain is a measure of the reduction in entropy (uncertainty) achieved by selecting a particular attribute (in this case, the query word) for splitting the data. The attribute that results in the highest information gain is selected as the splitting criterion. We can calculate the information gain of each candidate query word as follows:

1. Calculate the entropy of the current set of candidate words. The entropy is given by:

$$H(S) = -\sum P(i)log_2(P(i)) \tag{1}$$

   where $P(i)$ is the probability of selecting a word with label i from the set of candidate words.

2. For each query word, calculate the weighted entropy after splitting on that word. The weighted entropy is given by:

$$H(S|Q) = \sum \frac{|S_v|}{|S|} H(S_v) \tag{2}$$

   Where S_v is the subset of candidate words that match the query word, and $H(S_v)$ is the entropy of that subset.

3. Calculate the information gain achieved by splitting on that query word:

$$I_G(S,Q) = H(S) - H(S|Q) \tag{3}$$

4. The query word with the highest information gain is chosen as the splitting criterion at that node.

The second step is to determine when to stop expanding the decision tree and make a node a leaf. We initially used a pre-defined maximum depth of the tree. This prevented overfitting by limiting the complexity of the tree. The next approach we implemented is to stop splitting a node if the number of candidate words remaining is below a certain threshold (75 in the current scenario). This prevented overfitting by avoiding overly specific nodes.

To further prevent overfitting, we can implement pruning strategies such as reduced error pruning or cost complexity pruning. Reduced error pruning involves removing nodes that do not improve the accuracy of the tree on a validation set. Cost complexity pruning involves adding a regularization term to the information gain criterion to penalize overly complex trees.

1.2 Decisions Made

No puns intended, but we did make some critical decisions while attempting the assignment. Two decisions that stand out the most are as follows:

- **Global all_words**: From the beginning it was clear that the same words were being fed to the children nodes from the parent nodes. Hence, it was decided to altogether remove the list of words from each node and rather declare it as a global list which could be accessed by each node whenever required. This help us reduce the model size a bit.
- **History removal**: To further drastically reduce the model size, we decided to remove the history all together. This was done partly because there was no apparent need in the model for history to remain and occupied a ton of space without having a purpose. The only actual use of history we did was to check whether the given node was root or not and that could easily be done by checking if the root depth was 0 in the first place. Removing history helped us save $\tilde{5}0\%$ space on the model size.

1.3 Code Implementation

The code works on a list of words and generates a tree with nodes that perform a binary search using the queries. The code then trains the tree and returns the trained model.

- The `my_fit()` function takes in a list of words, initializes the decision tree, and calls the `fit()` function on the root node of the tree.
- The `DecisionTree` class contains the decision tree's root, which is initialized as None, the minimum leaf size, and the maximum depth of the tree. It also contains a `fit()` function that initializes the root node, trains it using all the words, and passes the minimum leaf size and maximum depth to the root node's `fit()` function.
- The `Node` class contains information about the node itself, including its depth, its parent node, its child nodes, its query index, its words index, and whether it is a leaf node. It also contains methods for generating a query, getting a child node based on a response, processing a leaf node, and calculating entropy.
- The `reveal()` function finds the intersection between the query and a word and returns a masked version of the word.
- The `try_attr()` function is called when entropy is calculated for a node. It takes a query string and a list of words and returns the entropy and the split dictionary for the node.

2 Solution to Question 2

The solution of this part can be found in the **submit.py** file in the uploaded zipped file.

3 Solution to Question 3

We tested our model on a number of dictionaries. They were all evaluated on the evaluation jupyter notebook provided on google colab. The benchmarks are listed below:

3.1 Provided Dictionary

Number of available words: 5,167

Benchmark	**dummy_submit**	**submit**
Training Time (in s)	0.2716	2.1325
Model Size (in B)	1966331	660009
Win Accuracy	0.8365	1.0
No. of Queries	10.06	4.07

3.2 English Dictionary

Number of available words: 400,036

Reference: @github: dwyl/english-words

Benchmark	**dummy_submit**	**submit**
Training Time (in s)	31.9386	146.7078
Model Size (in B)	148983972	56003937
Win Accuracy	0.6987	1.0
No. of Queries	12.0653	5.4315

3.3 Hindi Dictionary

Number of available words: 146,249

Reference: @github: gayatrivenugopal/Hindi-Aesthetics-Corpus

Benchmark	**dummy_submit**	**submit**
Training Time (in s)	3.5994	77.0371
Model Size (in B)	38417591	23663499
Win Accuracy	0.3238	0.9967
No. of Queries	14.0749	7.3816