CS771 : Assignment 2

Team Members

Anaavi Alok
Anurag Bohra
Dhwanit Balwani
Kartikeya Raghuvanshi
Prakhar Pratap Mall
Ujwal Kumar

1 Design Decisions

The logic used behind the code can be broken down into the following fundamental steps:

- First, we will fit the tree that calls for fitting the node (This is clear from the underlying structure of the data structure).
- In the `node_fit()` function, we first check whether the node is the leaf node, which is done by comparing max depth with the current depth and number of words under the node with min leaf node size. This is clearly not true for the first iteration.
- Now we process the node to split the tree at node by calling the function `process_node_()`. Where we first check whether the node is leaf or not. In the case of a root node we make a random word as the root word (in this case we chose the first word of the dictionary). If the node is not the root we move to the function `find_best_index()` to get the word index with minimum entropy (that splits in such a way that makes the entropy minimum) instead of choosing a random words from the full list of words.
- `find_best_index()` : This function first creates a list of all possible splittings with their entropy by using the function `calculate_word_entropy`. When we have this list, we just simply choose the word index with the minimum entropy.
- `calcualte_word_entropy()` : As described above the job of this function is to calculate the entropy when we use a particular word to split the given node. This is a accomplished by using a variable `mask` where each `mask` is made by the letters which are same in the query word and the chosen word. After going through all the words we have a list of masks. Now we go to `calculate_entropy()` to find the entropy using the formula as described in the next pointer.
- `calculate_entropy()` : The function has a list of masks each having a group of words. Now this function uses the formula of calculating the entropy by using the formula :

  $$\sum_{k\in[K]} \frac{n_k}{n} \cdot H(S_k)$$

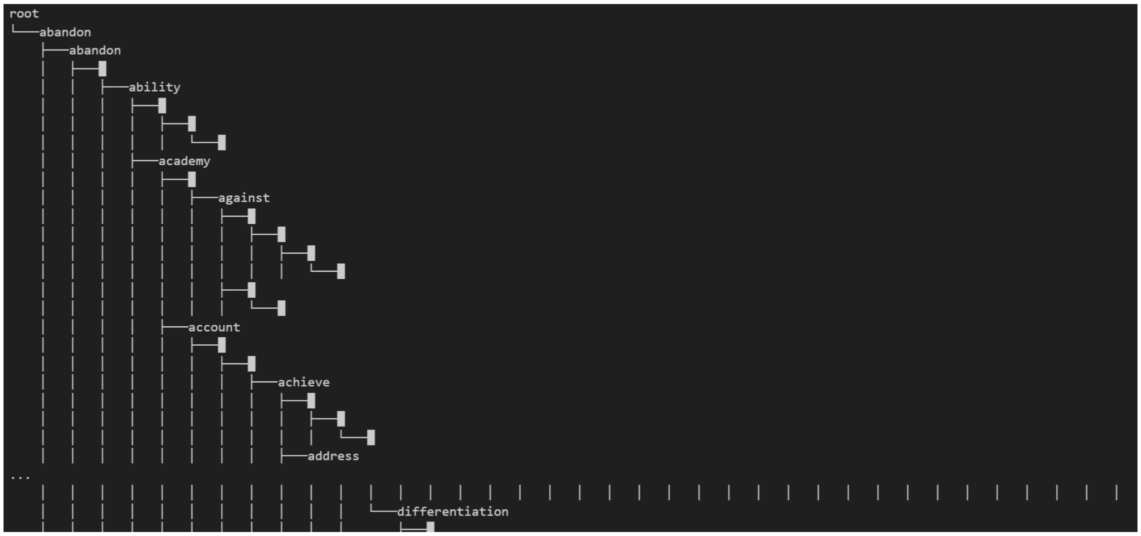
  where $n_k = |S_k|$ and S_k denotes the k^{th} subset, $S_1, S_2, ..., S_k$ being all the splits of the full dicitonary set and $H(S_k)$ is the entropy of the set S_k.

  We know $S_i \cap S_j = \phi$ if $i \neq j$ and $\cup_{k\in[K]} S_k = S$.
- Repeat the above steps untill each branch terminates at a leaf node.

2 Results

Train Time(s)	Model Size(bytes)	Accuracy(%)	Average number of queries
0.299	820998.0	1.0	5.144958389781304

Table 1: A sample run



```
root
└───abandon
    ├───abandon
    │   ├───█
    │   │   ├───ability
    │   │   │   ├───█
    │   │   │   │   ├───█
    │   │   │   │   │   └───█
    │   │   │   ├───academy
    │   │   │   │   ├───█
    │   │   │   │   │   ├───against
    │   │   │   │   │   │   ├───█
    │   │   │   │   │   │   │   ├───█
    │   │   │   │   │   │   │   │   ├───█
    │   │   │   │   │   │   │   │   │   └───█
    │   │   │   │   │   │   ├───█
    │   │   │   │   │   │   │   └───█
    │   │   │   │   ├───account
    │   │   │   │   │   ├───█
    │   │   │   │   │   │   ├───█
    │   │   │   │   │   │   │   ├───achieve
    │   │   │   │   │   │   │   │   ├───█
    │   │   │   │   │   │   │   │   │   ├───█
    │   │   │   │   │   │   │   │   │   │   └───█
    │   │   │   │   │   │   │   │   ├───address
...
    │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │   │
    │   │   │   │   │   │   │   │   │   │   │   └───differentiation
    │   │   │   │   │   │   │   │   │   │   │       ├───█
```

Figure 1: A sample tree

3 Code/Implementation

Link to code : http://home.iitk.ac.in/ ppmall20/neuralkrew.zip