CS771

April 5, 2023

Question 1

Our model's splitting criteria is based on Information Gain, Let's consider P is the parent node, and it has k children nodes for a given particular word, let's denote these children nodes as C_1, C_2, C_3,.....,C_k.

Let M be the total number of words in Parent node P, and M_i be the total number of words in the child node C_i and let $p_i = \frac{M_i}{M}$ be the portion of words in C_i.

Then entropy of node P is defined as:

$$H(P) = -\sum_{i=1}^{i=k}(p_i log_2(p_i))$$

The entropy of Child Node C_i is defined as:

$$H(C_i) = -\sum_{i=1}^{i=k}(p(\frac{d}{C_i})log_2(p(\frac{d}{C_i}))$$

where, $p(\frac{d}{C_i})$ is portions of words in C_i that belongs to class d.

The weighted average entropy of child nodes is defined as:

$$H(C) = -\sum_{i=1}^{i=k}(p_i H(C_i))$$

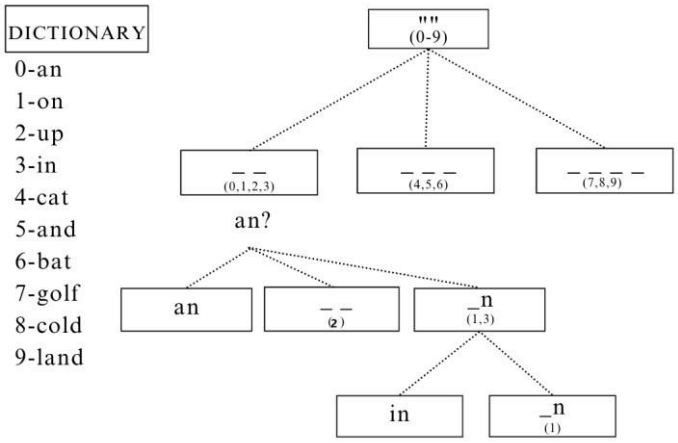
Therefore, Information Gain from splitting P using word w is defined as:

$$IG(P,w) = H(P) - H(C)$$

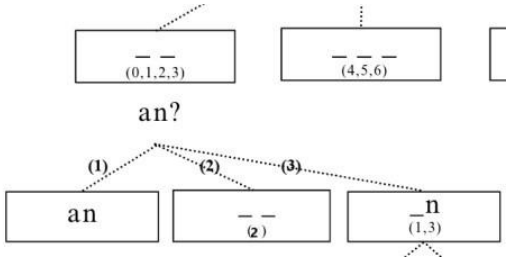
where C is a set of child nodes resulting from splitting P using the word w. Here, the word that maximizes the information gain is chosen as the query for the split. (or) We have to maximize $H(P)$.

For example, Let's choose [an, up, in, cat, and, bat, golf, cold, land, on] as the dictionary.

Given below is the decision tree diagram for the above dictionary.



- Firstly For the root node, we have to pass the empty query, at this step the decision tree will split the words based on word count.
- Now for example, if the given word to be checked is 'an', then we go to the node where the word count is 2. Let B be the set of words with a word count of 2 ([an, up, in, on]).
- After the above step, we calculate Information Gain for all the words in set B. Here, we should choose the word with maximum Information Gain as the query.In this case, the word 'an' will have maximum Information Gain.



- As shown in the above diagram in $path_1$, the word 'an' is the only word that matches completely in set B. This word will be stored in this node. here both letters are unmasked.
- In $path_2$, the word 'up' should be stored at this node as this is the only word with no similar words at the same index. So, both letters are masked.
- In $path_3$, In set B only 'on' and 'in' are left, and only the letter 'n' matches. So, it will be only character unmasked.

- From here we repeat the above steps and construct the decision tree recursively.

We took k number of words from all the words in the node-set for checking Information Gain and choosing the query among them. After checking various values of this hyperparameter we found $k = 75$ to be the optimal value which will minimize training time and also minimize queries.

In order to optimize the space, we are not storing history or any words but just storing the query index.

When maximum depth (15) is reached, the tree will stop expanding and the node will be converted into a leaf. If more than one word is present in that node, then the first word is chosen as the query.

In this case, pruning may not be necessary or beneficial for the following reasons:

- Limited complexity: The decision tree in this case is typically not very complex, as it only involves a limited number of possible letters at each step. Therefore, the risk of overfitting is low, and pruning may not lead to a significant improvement in performance.
- Lack of generalization: The goal of the decision tree, in this case, is not to generalize to new data, but rather to accurately solve the game for a specific word. Therefore, pruning may not be necessary to improve the model's ability to generalize.
- Incomplete information: In this case, the player only has partial information about the target word, which can make it difficult to accurately prune the decision tree.