CS771 Assigment-02

Aadi Singh **Aadya Dhir** **Aditya Thakur** **Nilesh** **Vedansh**

July 2024

Introduction

In this report, we evaluate the performance of a decision tree algorithm designed to predict words based on bigram lists. The algorithm incorporates a lookahead mechanism to improve precision by considering future splits. We discuss the design decisions, mathematical calculations, and performance metrics.

Design Decisions and Mathematical Calculations

Node Splitting Criteria

1. **Random Bigram Selection:**
   - Each node (`Node` object) selects a random bigram (`query`) from its current set of words (`all_words`).
2. **Bigram Presence Check:**
   - For each word in `my_words_idx` (the indices of words assigned to this node), the node checks if the randomly selected bigram (`query`) is present in the list of bigrams derived from that word.
3. **Splitting into True/False Responses:**
   - Based on whether the randomly selected bigram (`query`) is found in a word's bigram list, the node splits its assigned words (`my_words_idx`) into two subsets:
     - **True response:** Words where the bigram is present.
     - **False response:** Words where the bigram is not present.
4. **Handling Edge Cases:**
   - The splitting process handles cases where the randomly selected bigram might not effectively split the dataset into two meaningful subsets. In such cases, it's designed to still proceed with the split, potentially resulting in unbalanced or less effective splits.

Stopping Criteria:

- **Minimum Leaf Size (min_leaf_size)**:
  - If the number of words assigned to a node (my_words_idx) is less than or equal to min_leaf_size, the node is considered a leaf node. Further splitting is halted, and the node is processed as a leaf.
- **Maximum Depth (max_depth)**:
  - If the current depth of the node exceeds max_depth, no further splitting occurs. This prevents the tree from growing beyond a specified depth limit, which helps in preventing overfitting and improving generalization of the model.

Pruning Strategies for Decision Trees

- **Post-pruning (Reduced Error Pruning)**:
  - This technique involves growing the full tree and then pruning it back by removing nodes that do not significantly improve the accuracy of the tree on validation data. Typically, a separate validation set is used to evaluate the impact of pruning.
- **Cost-Complexity Pruning (Minimal Cost-Complexity Pruning)**:
  - This approach involves adding a complexity parameter (often denoted as α) to the tree-building process. It penalizes the tree for having too many nodes, thereby encouraging simpler trees. Trees are then pruned based on their cost-complexity measure.
- **Subtree Replacement**:
  - Instead of removing individual nodes, entire subtrees (branches) may be replaced with a leaf node or pruned based on their performance on validation data.
- **Depth Limitation**:
  - Restricting the maximum depth (max_depth) is a form of pre-pruning that prevents the tree from growing too complex. It indirectly helps in reducing overfitting.
- **Minimum Samples per Leaf**:
  - Similar to min_leaf_size, setting a minimum number of samples required to be in each leaf node (min_samples_leaf) can prevent the tree from splitting further if the resulting leaves would be too small.

Hyperparameters

The hyperparameters used in the decision tree algorithm are:

- **Minimum Leaf Size (min_leaf_size)**:
  - Controls the minimum number of words that a node must have to be considered a leaf. If a node has fewer words than min_leaf_size, further splitting stops, and the node becomes a leaf node.
- **Maximum Depth (max_depth)**:
  - Sets the maximum depth of the decision tree. If a node reaches this depth during the splitting process, further splitting stops, and the node becomes a leaf.

Performance Evaluation

We evaluate the performance based on training time, model size, testing time, and precision. The following results were obtained:

Metric	Value
Training Time (s)	0.01694
Model Size (bytes)	170583.0
Precision	0.6847
Testing Time (s)	0.68898

Table 1: Performance Metrics of Decision Tree Algorithm

Analysis

Potential Causes for Observed Performance

- **Training Time**: The lookahead mechanism increases training time due to additional computations for future splits.
- **Model Size**: The large model size is due to storing detailed information about each node and split.
- **Testing Time**: Testing time is influenced by the depth and complexity of the decision tree.
- **Precision**: The precision score indicates room for improvement, possibly through more sophisticated splitting and pruning strategies.

Conclusion

The decision tree algorithm with a lookahead mechanism demonstrates reasonable training and testing times, but the precision score suggests further refinement is needed. Future work will focus on optimizing splitting and pruning strategies to enhance performance.