Figure 1: Some examples of the response Melbot would give to Melbo?s queries on the words villager and universal. Note that if the query word is longer than the secret word, the extra characters are ignored. If the query word is shorter than the secret word, the remaining positions are padded with underscores. A character is revealed only if both the query word and the secret word have that character at the same location.

Problem Statement: Melbo has purchased a new toy to improve his vocabulary called the Melbot. The toy lets Melbo play a word-guessing game. There is a dictionary of N words that is known to both Melbo and Melbot.

In each round of this game, the following steps are taken:

1. Melbot chooses one of the words from the dictionary as secret say villager for this round.

2. Melbot tells Melbo the number of characters in that word by sending Melbo the string "_ _ _ _ _ _ _ _" (without the quotes)

3. The following steps are repeated until the round is terminated by either Melbo or Melbot.

(a) Melbo guesses a word from the dictionary by taking an index i between 0 and N ? 1 and sending it to Melbot as a query. For example, Melbo chooses the index 4972 that corresponds to the word violation.

(b) Melbot checks Melbo?s query to see if it is a valid one. If the query index is invalid i.e. i /? [0, N ? 1], then Melbot assumes that Melbo no longer wants to continue and terminates the round.

(c) If the query index is valid, Melbot checks if Melbo?s guess is indeed the secret word and if so, the round is terminated and the win count is incremented by 1.

(d) If the query word is not the secret word and Melbo has made too many queries in this round (the limit is Q = 15 queries per round), then Melbot terminates the round.

(e) If the query word is not the secret word but Melbo has not reached the query limit, then Melbot reveals to Melbo all characters that are common to the query word and the secret word (only if those characters are in the correct location as well).

For example, if the secret word is villager and the query word is violation, then Melbot would return the string "v i _ l a _ _ _" (without the quotes). See the figure for more examples.

Melbo plays N rounds of this game, once with each word in the dictionary. Melbo?s performance is judged based on the number of words guessed correctly within the query limit (the win count divided by N), the average number of queries asked per round, and other performance measures.

Note that at any point, Melbo can ask any word as a query as long as it is in the dictionary. It is not necessary that if Melbo is at a certain node in the decision tree, only one of the words that reached that node must be asked ? words that did not reach that node may also be asked if they help discriminate between words that reached that node.

Your Task: Develop a decision tree learning algorithm that can play this game. The algorithm will be tested on a secret dictionary that is different from the one provided.

The following tasks need to be accomplished:

1. Provide detailed calculations explaining the various design decisions taken to develop the decision tree algorithm. This includes the criterion to choose the splitting criterion at each internal node (which essentially decides the query word that Melbo asks when that node is reached), criterion to decide when to stop expanding the decision tree and make the node a leaf, any pruning strategies, and hyperparameters.
2. Write code implementing the decision tree learning algorithm. Use only the numpy library; other libraries like scikit-learn, scipy, skopt, etc., are not allowed.

The code must implement a method that takes a dictionary as a list of words and returns a trained decision tree as a model. The trained decision tree should be a tree object, and every node in that tree should be a node object.

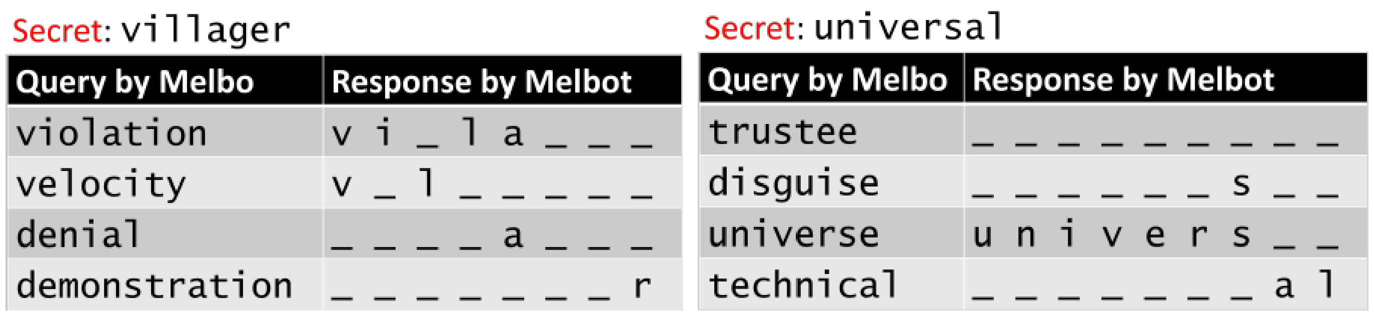
The algorithm will be evaluated on a different dictionary to check its effectiveness. The evaluation will be based on factors such as training speed, model size, number of queries made per round, and win rate on the secret dictionary.

Possible Solution Strategies: While standard information-theoretic entropy reduction algorithms can be used, there is room for experimentation:

1. Implementing lookahead can be a solid strategy. Instead of choosing the split that gives the maximum entropy reduction immediately, a better strategy might be to choose a split that will result in maximum entropy reduction two or three levels later. However, implementing large lookahead can increase training time significantly.
2. Another approach could be to minimize the number of queries it takes to correctly guess the word and maximize the win rate. For instance, if you have 100 words in a node at depth 5, find all subtrees that can be built from this node onward and choose the one that has the highest value of average win rate or the lowest average number of queries asked.
3. Experiment with splits that are not just based on entropy reduction or expected time to success but also offer a balanced split, considering this balance as a regularization term.

Figure 2: Greedy splitting can be suboptimal. In the example on the left, the split at the root was very nice (entropy 6.0) but it eventually led to a bad set of grandchildren (entropy 5.73). In the example on the right, the split at the root was not that nice (entropy 6.19 > 6.0) but the set of grandchildren it produced had lower entropy (entropy 5.19).

Figure 1:



Secret: villager

Query by Melbo	Response by Melbot
violation	v i _ l a _ _ _
velocity	v _ l _ _ _ _ _
denial	_ _ _ _ a _ _ _
demonstration	_ _ _ _ _ _ _ r

Secret: universal

Query by Melbo	Response by Melbot
trustee	_ _ _ _ _ _ _ _ _
disguise	_ _ _ _ _ _ s _ _
universe	u n i v e r s _ _
technical	_ _ _ _ _ _ _ a l