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

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 2.1 (Playing with the Melbot)**.** 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 check's Melbo's query to see if it is a valid one. If the query index is invalid i.e. $i \notin [0, N-1]$, 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 some other measures described below. Note that at any point, Melbo can ask any word as a query so long as it is in the dictionary. It is not necessary that if Melbo is at a certain node in the decision tree, then

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.

Before proceeding, **please take a look at the Google Colab validation code and the dummy submission file** `dummy_submit.py` provided with the assignment package to get an idea of how the above protocol works and how your evaluation would be done.

Note that Melbo can also terminate a round by setting the `is_done` flag but that is automatically done when Melbo reaches the leaf of the decision tree. There is no way for you to specify this flag in your solution. When you are writing a code and wish to terminate a round, you should instead send an illegal index to signal termination of the round.

Your Data. We have provided you with a dictionary of $N = 5167$ words. Each word in the dictionary is written using only lower-case Latin characters i.e. `a - z`.

Your Task. You need to develop a decision tree learning algorithm that can play this game. However, note that your algorithm will be tested on a secret dictionary that we have not revealed to you. More on this later. The following enumerates the 2 parts to the question. Part 1 needs to be answered in the PDF file containing your report. Part 2 needs to be answered in the Python file.

1. Give detailed calculations explaining the various design decisions you took to develop your 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 etc. (10 marks)

2. Write code implementing your decision tree learning algorithm. You are not allowed to use any library other than numpy. This means that even use of scikit-learn is prohibited. Use of other libraries such as scipy, skopt, etc is also forbidden. Submit code for your chosen method in `submit.py`. Your code must implement a `my_fit()` method that takes a dictionary as a list of words and returns a trained decision tree as a model. The trained decision tree as a model should be a tree object. Every node in that tree should be a node object. There is no restriction on what attributes the tree object or the node objects may have and what methods those classes implement (i.e. feel free to implement your own Tree and Node classes) but the Node class must implement at least 2 methods:

   (a) Every non-leaf node should implement a `get_child()` method that takes a response and decides which child node to move to.

   (b) Every node (leaf as well as non-leaf) should implement a `get_query()` method that tells what query Melbo should ask when at that node. For leaf nodes, this would be the final query in that round after which the round will be terminated.

   We will evaluate your method on a different dictionary than the one we have given you and check how good is the algorithm you submitted (see below for details). **Please go over the Google Colab validation code and the dummy submission file** `dummy\_submit.py` to clarify any doubts about data formats, protocol etc. (30 marks)

Part 1 needs to be answered in the PDF file containing your report. Part 2 needs to be answered in the Python file.