CS771 Assignment 2

Supervised Learners

Solution 1

Algorithm for building and using a decision tree with bigrams

Initialization

- Generate all unique bigrams for the given words
  - Extract bigrams from each word
  - Return a sorted list of these unique bigrams
- Create the root node of the tree with all the words and the generated bigrams.

Tree Structure

- The tree is built by recursively splitting the words based on the bigrams.
- Splitting criteria for Bigrams and Words:
  - At each node, the list of bigrams is split into two approximately equal halves, creating left and right child nodes. This splitting process continues until there is only one bigram left or no words are left to split.
  - If the number of bigrams is less than or equal to 1, or if there is only one word, assign the words to the node and continue to the next node.

Maximum Nodes

- Each split potentially creates two child nodes from one parent node. If every node creates two child nodes and this continues until the smallest possible segment (one bigram), the maximum number of nodes can be approximated by considering a binary tree structure.
- Given this, let's estimate the maximum nodes:
  - The tree starts with B bigrams at the root.
  - The depth d of the tree is roughly $\log_2(B)$, where **B** is the number of unique bigrams.
  - Thus, the maximum number of nodes N in the tree can be approximated by:
    $\mathbf{N} = 2^{\log_2 B+1} - 1 \approx 2\mathbf{B} - 1$
- In our case number of nodes generated for `dict_secret` is **869**.

Helper Function

- generate_bigrams: Extracts unique bigrams from a list of words
- split_words_by_bigrams: Splits words into those containing any of the given bigrams.
- _count_nodes: Recursively counts the number of nodes in the tree

Working of the Algorithm

- After getting the list of 5 lexicographically sorted bigrams we pass the bigrams into the decision tree one by one and zero-in on the words containing that bigram.
- After each iteration the word set at the node is reduced to the previously deduced words from previous iteration.
- Another approach could be to find 5 different sets of words from the same decision tree and at last return the top 5 words from the intersection of those sets.
- At last we are left with a set of words and the function `my_predict` returns first 5 elements from that set.

TEAM MEMBERS:

Prithvi Mehta	**220819**	**prithvim22@iitk.ac.in**
Aditi Singh	**220062**	**aditisi22@iitk.ac.in**
Abhishek Srivastava	**220051**	**srivabhi22@iitk.ac.in**
Aayush Gautam	**220020**	**aayushg22@iitk.ac.in**
Guguloth Pavani Priya	**220415**	**gppriya22@iitk.ac.in**