CS771 ASSIGNMENT-2 REPORT

1 Introduction

This document outlines the design decisions and calculations behind the development of a machine learning algorithm using a Decision Tree Classifier for bigram-based word prediction. The task of predicting words based on given bigrams involves processing textual data where each word is represented by a sequence of lexographically sorted bigrams. This study addresses the challenge of efficiently predicting words using only the first five unique bigrams of each word. The bigrams are extracted in a manner that ensures lexicographical order and removes duplicates, facilitating a streamlined input for prediction models. By leveraging these preprocessed bigrams, the prediction system aims to accurately suggest potential words associated with a provided set of bigrams, optimizing both computational efficiency and predictive accuracy in natural language processing applications.

2 Approach For Solving Problem

The problem to predict the words based on given bigrams as an input where bigrams are geographically sorted and considered only first five creating a challenge in predicting the words that are having same set of bigrams after the preprocessing.
The problem can be solved using various ways either by simply using a hash map which maps the words with their bigrams and given a bigram as input it returns a list of words having that bigram or we can use a Decision Tree algorithm to solve it.

2.1 Choosing the Algorithm

The *Decision Tree Classifier* was chosen due to its simplicity and interpretability. Decision trees are non-parametric models that work well with categorical data and can handle complex decision boundaries expected in text classification tasks such as bigram-based word prediction.

2.2 Bigram Calculation

The function `extract_bigrams(term)` is designed to generate bigrams from each word. Bigrams are sequences of two consecutive characters from a word.

This is useful as bigrams can capture the local structure and context within words, which are essential for distinguishing between different terms.

2.3 Preprocessing and Feature Extraction

The `organize_words(terms)` function organizes words by their bigrams and collects all unique bigrams. It limits the number of bigrams to the first 5 unique sorted bigrams for each word to manage the feature space size and focus on the most significant bigrams. The choice of 5 bigrams is a heuristic to balance between capturing sufficient information and avoiding overfitting with too many features.

2.4 Label Encoding

LabelEncoder is used to convert the target words into numeric labels. This is necessary because the decision tree implementation in *scikit-learn* requires numeric input for the target variable. This encoding helps in transforming categorical word labels into a format suitable for the classifier.

2.5 MultiLabel Binarization

MultiLabelBinarizer is used to transform the set of bigrams (features) into a binary matrix. This binary representation is suitable for the decision tree algorithm, which can then learn to split based on the presence or absence of specific bigrams. This approach helps in handling the multi-label nature of the input where each word can be associated with multiple bigrams.

2.6 Training the Model

The `train` method of the `SequenceModel` class fits the *DecisionTreeClassifier* on the transformed bigram features and the encoded word labels.

- `DecisionTreeClassifier(criterion='entropy')` specifies the use of entropy (information gain) as the criterion for splitting nodes. Entropy measures the impurity in a dataset; hence, selecting splits that reduce entropy helps in building a tree that makes good predictions.
- Entropy is chosen because it often provides better splits when dealing with categorical data compared to other criteria like Gini impurity.

2.7 Entropy Reduction and Information Gain

The entropy H of a dataset S is given by:

$$H(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

where c is the number of classes and p_i is the probability of class i.

Information Gain

The Information Gain IG when splitting a dataset S into $S_1, S_2, \ldots, S_n$ based on an attribute A is calculated as:

$$IG(S, A) = H(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} H(S_i)$$

where:

- $H(S)$ is the entropy of the original dataset S.
- $H(S_i)$ is the entropy of subset S_i.
- $\frac{|S_i|}{|S|}$ is the weight of the subset S_i relative to the original set S.

This criteria of splitting the words in children nodes is used to make a decision tree classifier so that it can make splits which can reduce entropy maximum. This Algorithm is Known as the **ID3 algorithm** which chooses the split which gives maximum information gain.

2.8 Stopping Criterion and Pruning

In this implementation, default settings of *DecisionTreeClassifier* are used, which means the tree grows until all leaves are pure or until all leaves contain less than the minimum samples required to split. Pruning strategies (e.g., setting a maximum depth, minimum samples per leaf) are not explicitly mentioned in the provided code but can be incorporated based on the problem's requirements and to prevent overfitting.

2.9 Prediction

The `predict` method transforms the input bigrams into the same binary matrix format using *MultiLabelBinarizer* and predicts the word labels using the trained decision tree model. The predicted numeric labels are then inverse transformed to obtain the original word labels.

3 Summary of Hyperparameters and Design Choices

- **Bigrams Limitation**: Each word is limited to the first 5 unique bigrams.
- **Entropy Criterion**: Used for selecting the best splits in the decision tree.
- **No Explicit Pruning**: Default settings are used; however, pruning can be added if needed to manage overfitting.

These design decisions ensure the model is simple, interpretable, and performs well on the given task by capturing the essential structure within words through bigrams and using a decision tree for classification.

4 Results

The implemented word prediction model, utilizing geographically sorted bigrams with duplicates removed and limited to the first five bigrams, achieved notable performance metrics. The model demonstrated a high **precision score of 0.97**, indicating its strong accuracy in predicting the correct words based on the given bigrams. Furthermore, the training process was highly efficient, with a training time of only 0.0097 seconds, underscoring the model's computational efficiency. These results highlight the effectiveness and practicality of the proposed approach in word prediction tasks.

The Team

Name	Email	Roll Number
Akash Kumar Gupta	akashkg22@iitk.ac.in	220095
Dhruv Bansal	dhrubb22@iitk.ac.in	220359
Raunak Raj	raunakr22@iitk.ac.in	220876
Dhruv Mittal	mdhruv22@iitk.ac.in	220364
Ayush Shrivastava	ayushsri22@iitk.ac.in	220272
Ajay Singh	ajaysi22@iitk.ac.in	220090