CS771 : Introduction to Machine Learning
Assignment - 2

Anubhav Gyanendra Singh
220179
anubhavgs22@iitk.ac.in

Agam Pipersenia
220086
agamp22@iitk.ac.in

Ashwin Chaubey
220245
ashwinc22@iitk.ac.in

Dhruv Mittal
220363
dhruvm22@iitk.ac.in

Suman Kumar Maharana
232090906
sumanm20@iitk.ac.in

Yash Chauhan
221217
yashc22@iitk.ac.in

Abstract

In this project, we address the problem of word sequencing using a machine learning approach to predict words from given bigrams. The task is inspired by genetic sequencing techniques where the aim is to reconstruct sequences from overlapping fragments. We employ a decision tree classifier to tackle this problem, leveraging its ability to handle non-linear relationships and categorical data efficiently. We preprocess the words by extracting, deduplicating, sorting, and truncating their bigrams to create feature vectors. These vectors serve as input to our decision tree model. The classifier is trained to predict words based on the presence of specific bigrams. Hyperparameters such as maximum tree depth, minimum samples per split, minimum samples per leaf, and minimum impurity decrease are carefully chosen to balance model complexity and performance.

Our approach is validated on a provided dictionary of words, ensuring that the model generalizes well to unseen data. The evaluation metric is precision, which measures the accuracy of the predictions while penalizing for multiple guesses. The developed model demonstrates robust performance, achieving high precision in predicting words from their bigram sets. This work demonstrates the feasibility of applying decision tree algorithms to sequence reconstruction problems in natural language processing, with potential applications extending to various fields such as computational biology and text analysis.

1 Design Decisions and Implementation Details

In this section, we detail the design decisions and implementation steps taken to develop the machine learning algorithm for predicting words from bigrams. This involves the choice of algorithm, feature extraction, model training, and the criteria used to optimize the decision tree.

1.1 Algorithm Choice

We selected a decision tree classifier for this task due to its capability to handle categorical data and non-linear relationships. Decision trees are also interpretable, making it easier to understand the decision-making process based on bigram presence.

1.2 Feature Extraction

The primary challenge was to convert words into a format suitable for the decision tree. The steps involved are:

1. **Extract Bigrams**: For each word, we extract all possible bigrams. For example, the word "area" produces the bigrams ['ar', 're', 'ea'].
2. **Deduplicate and Sort**: The bigrams are then deduplicated and sorted lexicographically. This ensures a consistent order of bigrams, which is crucial for creating comparable feature vectors.
3. **Truncate**: We retain only the first 5 bigrams to comply with the problem constraints. If a word has fewer than 5 bigrams, all its bigrams are used.

1.3 Vocabulary Creation

To convert words into feature vectors, we first create a vocabulary of unique bigrams from the training dataset. Each bigram is assigned a unique index. The vocabulary allows us to map bigrams to specific positions in the feature vectors.

1.4 Feature Vector Representation

Each word is represented as a binary vector of length equal to the size of the vocabulary. The presence of a bigram in a word sets the corresponding position in the vector to 1; otherwise, it is 0. This binary representation is fed into the decision tree.

1.5 Model Training

The decision tree is trained using the extracted feature vectors and corresponding words. The training process involves:

1. **Splitting Criterion**: We use entropy as the splitting criterion. Entropy measures the impurity in a set of training examples, and minimizing entropy at each split maximizes information gain.
2. **Stopping Criteria**:
   - Maximum tree depth (`max_depth=5`) to prevent overfitting.
   - Minimum samples per split (`min_samples_split=2`) to ensure that splits are made only when there is a sufficient number of samples.
   - Minimum samples per leaf (`min_samples_leaf=1`) to allow leaves to contain at least one sample.

1.6 Pruning Strategy

To avoid overfitting, we use the `max_depth` parameter as a form of pre-pruning. This ensures that nodes are split only if they result in a significant reduction in impurity.

1.7 Prediction Process

During prediction, we convert the given list of bigrams into a binary feature vector using the same vocabulary. The trained decision tree classifier then predicts the most likely words. The process is as follows:

1. **Convert to Feature Vector**: Convert the input bigrams into a binary vector.

2. **Predict with Decision Tree**: Use the decision tree to predict probabilities for each class (word).
3. **Select Top Guesses**: Identify the top words with the highest probabilities. If the list contains more than 5 guesses, only the first 5 are considered.

1.8 Evaluation

The model's performance is evaluated using precision. Precision is calculated as the inverse of the number of guesses needed to correctly predict a word. The final precision is the average precision across all test points.

This structured approach ensures that the decision tree classifier is both accurate and efficient in predicting words based on bigram features while adhering to the constraints and addressing the potential ambiguities introduced by bigram truncation.

2 Python Code Implementation

*Zipped **Solution** to Assignment 2*