LAB 2 Report

Course:- FOUNDATIONS OF ARTIFICIAL INTELLIGENCE
Tutor- Niyazi Sorkunlu.
Author- Rishabh Manish Sahlot(rs3655@rit.edu)

Problem Definition

Given a 15-word vector/ sentence we need to show which language does this belong to from Dutch & French

Dataset

Text data was obtained from Gutenberg project, it is present in the Dutch & English folders.

We use dataset_maker.py to make testing & training data.

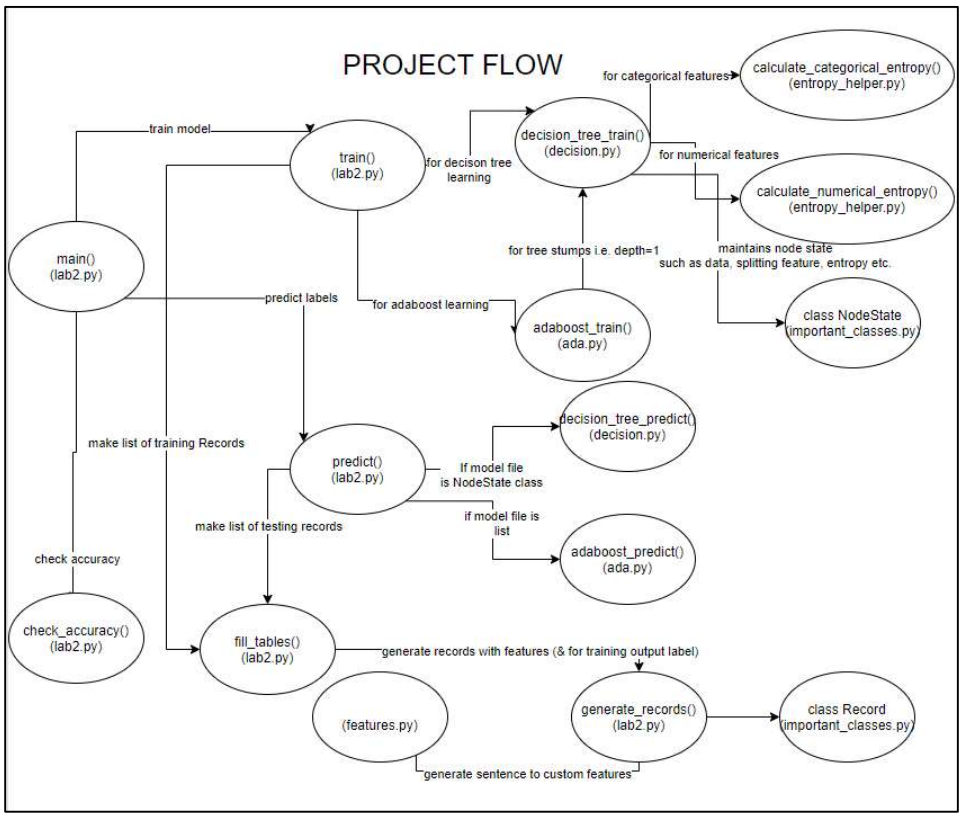
For making training dataset,

Type ' `python3 dataset_maker training <source_path> <language_of_source_file(en/nl)> <output_path>` '

For making testing dataset,

Type ' `python3 dataset_maker training <source_path> <language_of_source_file(en/nl)> <output_path_of_test_cases> <output_path of test_labels>` '

Additionally, I make use of dutch_stopwords.txt & dutch_stopwords.txt for my feature formation. The path to them was hardcoded in the start of train & predict methods in lab2.py. Please, change it accordingly to run.



LAB 2 Report

Course:- FOUNDATIONS OF ARTIFICIAL INTELLIGENCE
Tutor- Niyazi Sorkunlu.
Author- Rishabh Manish Sahlot(rs3655@rit.edu)

Features

Feature	*Description*	*Feature Type*	*Reason for use*
Dutch stopword checker	Checks from a list of dutch stop words(dutch_stop_words.txt required).	Categorical	Chances of them being present is high & they indicate for Dutch language
English Stopword checker	Checks from a list of english stop words(english_stop_words.txt required).	Categorical	Chances of them being present is high & they indicate for English language
Length of longest word	Check for length of longest word in each sentence.	Numerical	Dutch contains compound words which end up being longer
Weird ascii char checker	Presence of non-english like alphabet symbols.	Categorical	Dutch contains extra alphabets
Longest streak consonants	Length of Longest continuous sequence of consonants.	Numerical	English or dutch will surely have longer consonant.
Longest streak vowels	Length of Longest continuous sequence of vowels.	Numerical	English usually has less than 3 vowels at a time.
Percent frequency of c	Percentage of c from the all alphabets used.	Numerical	Dutch has lower frequency of c
Percent frequency of k	Percentage of k from all alphabets used.	Numerical	English has lower frequency of k
Average word length	Average length of word For each sentence.	Numerical	Due to compound words average word length is higher in dutch.
Average letter distribution	Takes the ordinal values of alphabets, sums them up & averages them.	Numerical	It helps in finding a estimate of distribution

They are defined in features.py

For categorical features, my code will allow multiple categories for testing, for numerical features my code will sort by feature value & then find optimal split value.

Categorical features are never considered again down the tree, but for numerical I have allowed for new split threshold for the corresponding data values to be made.

For future scope we can make use of monogram, bigram & diagram distributions and do a chi square goodness of fit test for comparing observed & expected frequency so as to get an estimate of the p-values for rejecting null hypothesis for both languages. A test

implementation (not used in current features) of how it will look like is present in features.py (monogram chi test).

Decision Tree Learning

The decision tree allows for max depth cutoff as well as max entropy cut off (i.e. lower prefered) which is set in set in lab2.py in train().

My decision tree works on two different types of data, numerical and categorical.

It has been implemented as a stack. Given a minimum accuracy(upto which to be trained) for the dataset, I estimate the cutoff value for entropy. The algorithm can be seen as a bfs.

For numerical data I first perform a sorting operation, post which it could possibly be split into two categories-less than a splitting value and greater than the splitting value. Where each split value is the mean of feature value of successive records in the sorted data.

Information Gain, split value $=$ arg $\_max_{Information\ Gain}\ H(Target) - \frac{i * Entropy(Output|feature\ value < split\ value) + (Dataset_{size} - i) * Entro \quad (Output|feature\ value \geq split\ value)}{Dataset\_size}$

Where i from 1 to data_set size-1.(First & Last no need to check since they don't make a split.)

For categorical data ,

Information Gain $=$ arg $\_max_{Information\ Gain}\ H(Target) - \sum_{each\ value\ of\ the\ feature} P(feature\ value) * Entropy(Output|feature\ value)$

We consider of all **these** (present in the that tree node) features for selecting the splitting feature.

For predicting I have iteratively traversed the model for each feature for the testing record & return the plurality value for that leaf node(There are no 0 record nodes created, so we don't need to check for parent).

For training the code can be run as

'python3 lab2.py training <training_source_path> <hypothesis_output_path> <output_path_of_test_cases> dt <dutch_stopwords_path> <English_stopwords_path>'

For predicting labels

'python3 lab2.py testing <testing_model_path> <testing_data_path> <predicted labels path> <dutch_stopwords_path> <English_stopwords_path>'

Boosting

For adaboost I make use of n(=30, set in lab2.py in train()) estimators. I make use of decision tree of depth 1 for each weak classifier. Later we could improve by taking

decreasing amount of depths. I have put an additional smoothening value of 1/data_set size & the formula for significance becomes significance $= 0.5 * log_{10}(\frac{1-error_{rate}+epsilon}{error_{rate}+epsilon})$. This solves the error_rate $= 0$ condition.

Algorithm for training.

Repeat For i[th] estimator:
- Find a decision tree stump of depth 1
- Find the count of erroneous records for each split branches
- Calculate error rate by adding these count and dividing by dataset size.
- Calculate significance alpha.
- Now initial sample weight value was 1/data_set size.
- For each record in the current data update is value as
  - W = W * e^{-alpha} , for correct values
  - W = W * e^{alpha} , for wrong values
- Normalize these weights.
- Find the cumulative weights.
- To make the new dataset, for each ith record (i goes till initial data_set size),
  - Randomly select a value between 0 & 1. Then use binary search to find the the record with smallest cumulative weight greater than that value.

Algorithm for predicting.

Set y = 0
Repeat For i[th] estimator:
- Find decision tree prediction of stump
- For English add the stump's significane value to y & substract it otherwise.

If y greater than 0 return English, else return French.

For training data

'python3 lab2.py training <training_source_path> <hypothesis_output_path> <output_path_of_test_cases> dt <dutch_stopwords_path> <English_stopwords_path>'

For predicting labels

'python3 lab2.py testing <testing_model_path> <testing_data_path> <predicted labels path> <dutch_stopwords_path> <English_stopwords_path>'

Accuracy

	Decision Tree	Adaboost
Accuracy limit for training	0.99	-
No of estimators	-	30
Test Size	1659	1659
Accuracy	95.05	94.87
False 'en'	43	43
False 'nl'	39	42