UNIVERSITÀ
di VERONA

LANGUAGE CLASSIFICATION USING N-GRAM ENCODING WITH DIFFERENT MACHINE LEARNING CLASSIFIERS

Master's degree in computer engineering for Robotics and Smart Industry

Subject:
Machine Learning & Artificial Intelligence

Submitted to:
Vittorio Murino, Cigdem Beyan & Andrea Avogaro

Submitted by:
Abdul Basit

ACADEMIC YEAR 2024/2025
VERONA UNIVERSITY, ITALY

Index

9. Model Evaluation

1. Evaluation Metrics (Accuracy, F1-Score)
2. Confusion Matrix
3. Model Evaluation

10. Model Testing on Unknown Samples

1. Query Sample Encoding
2. Prediction and Evaluation
3. New Sample Prediction

11. Conclusion
12. References

Decleration,

This report was originally submitted as a joint project by myself, Abdul Basit, and my former group-mate Muhammad Nouman. However, due to unforeseen circumstances, my group-mate has withdrawn from the Master's program and will no longer be participating in the project presentation. Given this situation, I am resubmitting the report with my name only and will be presenting the project individually in the upcoming appeal.

1. INTRODUCTION

Project Overview

The goal of the task is to build a language detection system on text data. The system is designed to preprocess the input text, encode it into n-grams, and then employ Ridge Regression as well as Decision Tree classifiers for the classification of the language in which the mentioned text was written. These models were used to evaluate how effectively a language can be detected based on its textual features. The project aims to demonstrate the capabilities of n-gram encoding in capturing the contextual and sequential information within textual content, which is crucial for accurate language identification. By comparing the performance of Ridge Regression and Decision Tree classifiers, the project seeks to highlight their respective strengths, limitations, and potential use cases in the field of language classification.

Motivation and Rationale

The language classification plays an important role in some practical applications such as content filtering, machine translation, NLP(Natural Language Processing) and multilingual text procession. Since people speak so many different languages all over the world, it is crucial to have an effective and scalable language classifier. In this project n-gram encoding is used to capture the information of context and sequence within textual content for language identification along with Ridge Regression, Decision Tree classifiers. By comparing these models, the project seeks to highlight their strengths, limitations, and potential use cases.

2. STATE OF THE ART

Ridge Regression

Ridge Regression is a linear regression model which applies L2 regularization to avoid overfitting. It is good for classification of languages in large feature space like n-gram encoding. How it helps reduce multicollinearity in independent variables.

Decision Tree

The Decision Tree is a non-parametric model that naturally organises the data by making decision rules based on the features. This is simple to understand and very interpretable, but you also risk overfitting here (especially with the kind of high-dimensional data). In this project, a Decision Tree classifier is used to learn the hierarchical decision-making process that takes place during language classification.

3. LOAD DATA

Data Extraction and Selection

This work has been implemented with the help of Leipzig Corpora Collection dataset. Each file corresponds to a language, each containing multiple sentences. As a trade-off between being computationally expensive and the performance of models, considering that there exist 200 sentences per language, we choose to compute all samples on this bilingual sub-corpora. This extraction ensures a manageable dataset size while maintaining sufficient data diversity for effective model training.

4. DATA CHUNKING AND PREPARATION

Text Processing

Sentence reading from text files in language folders at the preprocessing phase. Language folders similarly hold copious files, of which I pulled a fixed number of sentences. These File tools are used to store the splited setences so that they can be processed further. This step helps in making the dataset evenly distributed across languages, eliminating language bias during model training.

Error Handling

Errors that can happen during text processing like file access issues, missing files and corrupted data. It is necessary to store this error in the system, where a efficient mechanism has been built up which would record all such occasional errors and keep data consistent. This logging mechanism logs the errors which occurred during file operations, this is very helpful for identification while giving support to troubleshoot and at last we know that data extraction happens in an accurate way or not.

5. ADVANCED DATA CLEANING AND PREPROCESSING

Initialization

This stage begins with the initialization of several tools essential for text preprocessing, such as lemmatizers for reducing words to their base form, stopwords removal tools to filter out common words that do not contribute to language discrimination, and language mappings to handle multilingual datasets effectively.

Text Cleaning Functions

A series of text cleaning functions are applied to ensure that the input text is free from noise. These functions include:

- **Expanding Contractions:** Converts shortened forms (e.g., "don't") to their full forms ("do not").
- **Removing Special Characters:** Eliminates punctuation and symbols that do not carry linguistic meaning.
- **Removing URLs, HTML Tags, and Accents:** Cleans up the text by removing non-text elements.
- **Lemmatization:** Reduces words to their base or root form, enhancing model training by standardizing word forms.

Text Processing Functions

The cleaned text will then be used to construct the n-gram features needed for language classification. For each language we would tokenize the text into sentences and words, extract n-grams to capture information specific with that particular language.

Data Loading and Cleaning

The cleaned files are loaded and any final issues found like repeated sentences or wrong tags get solved. All samples are labelled, partly based on their language through automated labelling practices like the one we will describe.

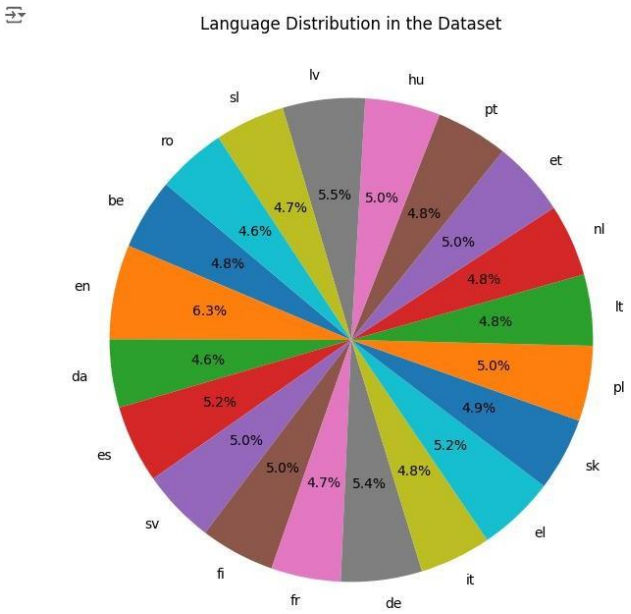
Data Display and Analysis

We are to verify if the data preprocessing step has been effective and check that in cleaned up data. This includes showing you example sentences, verifying that n-grams are present and making sure the cleaned data has all of the linguistic features for which classification.

6. DATA VISUALIZATION

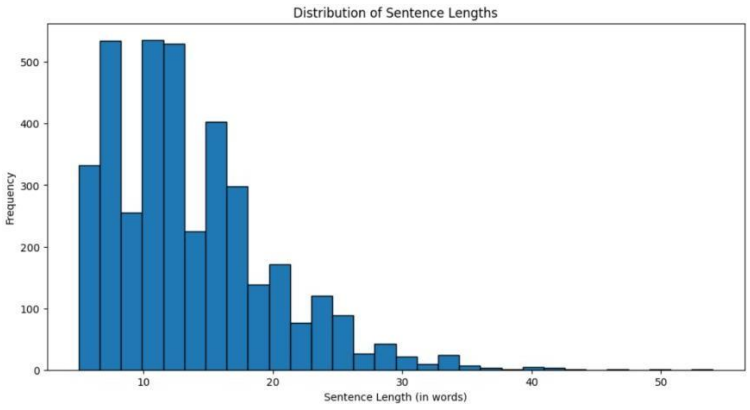
Language Distribution

This pie chart shows the percentage of different languages in this dataset, giving a snapshot picture about how composite is my dataset. This feed forward neural network helps to flag biases that might effect training of the model.



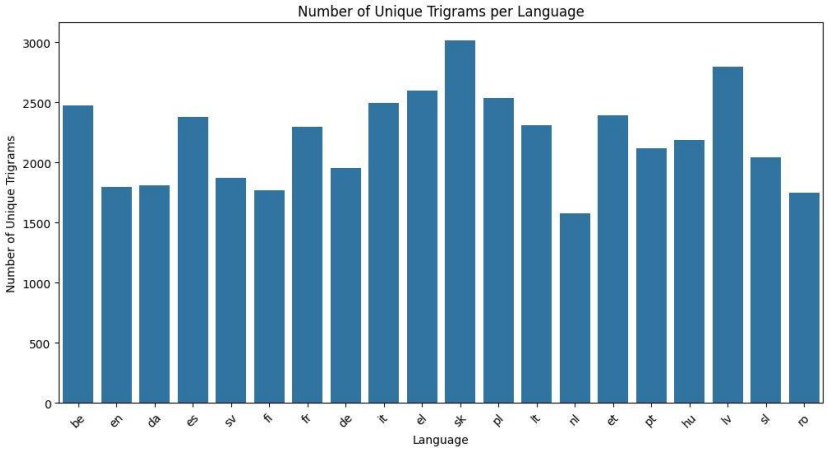
Sentence Lengths

We can use a histogram to show how sentence length varies across the dataset in word count. This visualization makes exploration of sentence level stuff easier and put that into context how some kind of variation will impaqct model performance.



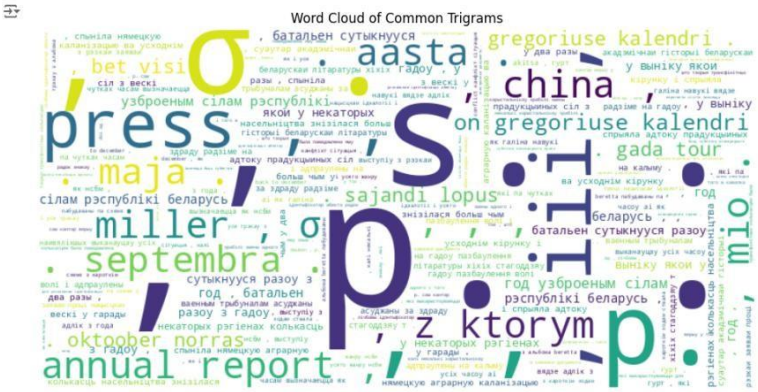
Unique Trigrams per Language

Bar chart that shows the count of unique trigrams found, per language. Importantly, this is also what those n-grams capture that makes them useful to distinguish between languages.



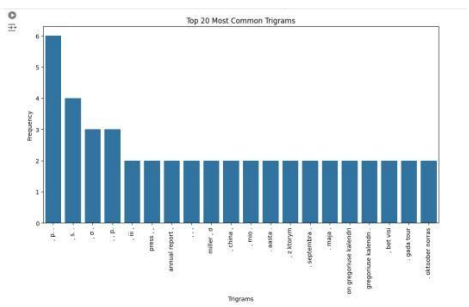
Word Cloud of Common Trigrams

This section shows the most popular trigrams in entire dataset which can make clear what type of word combination occurs more and frequently used.



Trigram Frequency Distribution

This section plots the top three most common click sequences of 3 words, allowing us to understand some popular phrases or word sequence that our publishers are using.



7. FEATURE ENGINEERING

Sentence Encoding into High-Dimensional Vectors

Algorithm: n-gram encoding for each sentence and then encode it in high dimensions vector. This step is a crucial one which converts textual data into numeric form and enables machine learning models to learn how complex patterns look like in different languages.

Vectorize the Text Data with n-grams

The specific process is creating High-Dimensional (HD) vectors for tri-grams — three consecutive characters-by using a training set.

- Define Dimensions: Specifies vector lengths of 100, 1K and10K for the HD vectors.
- Initialize Storage: Creates a dictionary to keep the HD vectors of each dimension.
- Generate HD Vectors: Create a random vector for each trigram, convert them into binary (+1 or -1), store it in the dictionary.
- Extract Trigrams: CountVectorizer is used to extract trigrams from the training data.
- Generate Vectors: Using the same trigrams above, where all dimensions in vector generation are applied by HD.

This code encodes sentences into HD vectors, with fixed trigrams. Tasks carried out by it are :

- Encode Sentence: This function outputs hash vector of sentence, given a text and an instance to the class HDVector. It tokenizes sentences into trigrams Tokenizing Agglomerates The respective_hash_vectors Normalize Step
- Encode Training Data: That goes through each sentence in the training set and creates HD vectors of varying dimensions for them, which are stored as keys to a dictionary.

Label Encoding

The labels are converted from language to a number so that the model can learn better. This step makes valid language categories comprehensible by the models.

8. MODEL TRAINING

Training Ridge Regression Models

Training Process

This section of the code converts sentences to HD vectors, processes labels and trains RidgeClassifier models for each dimension, saving them such that they could be used later.

9. Encode Sentences: Change the Sentence into HD Vector By Breaking it Into Trigams So That Each Trigram Will Corresponding to 100,000 Dimensions Ones and Aggregate Their Respective HashVector To Get Feature Vectors With Particular Dimensions.
10. Encode Training Data: This is done by going through the encoding process for all training sentences over each of those axis separately, and storing these encoded feature vectors in a dictionary
11. Label Encoding: Using Label Encoder, label encoding is performed to convert categorical labels into numerical values in order for the stages of model training.
12. Train Models: Train Ridge Classifier models using the shuffled training data generated in each other dimension.
13. Store Models: Save each trained RidgeClassifier model in a dictionary so we can use it later.

Training Decision Tree Models

The encoded data is also used to train Decision Tree models The model we use is Decision Tree, a non-linear classifier able to capture intricate decision rules that lead to the differentiation of one language with another by looking at their n-gram varied patterns.

14. MODEL EVALUATION

Evaluation Metrics

The performance of both models—Ridge Regression and Decision Tree—is evaluated using several metrics:

- **Accuracy:** The number of instances the language was predicted correctly divided by total no. of class
- **F1-Score**: The weighted average of Precision and Recall produce one in a single value for performance measure.

The Result can be Found on the project link
https://github.com/AbdulBasit0406/AI_ML_Project/blob/main/Language_Classification.ipynb

Confusion Matrix

We then generate confusion matrix on both models to provide some insight into the classification of each model. The matrices give information on the number of true positives, false positives, true negatives and false negatives for each model that we can use to understand where our models are doing well or extremely bad.

The Result can be Found on the project link
https://github.com/AbdulBasit0406/AI_ML_Project/blob/main/Language_Classification.ipynb

Model Evaluation

The pipeline transforms the text samples into high-dimensional vectors and encodes them, apply Ridge regression for predictions, evaluate performance using accuracy and F1 scores,and predict on new examples.

- Encode Query (a sample query processor): The system tries to identify the language of an input text, perform whatever pre-processing is necessary and then encode this into high-dimensional vectors.
- Label Encoding Step: Transforming training and testing labels into integer classes using LabelEncoder and saves in y_train_encoded and y_test_encoded.
- Encode Test Data: Test dataset sentence to HD vectors on all axis are stored in X_test_encoded ·
- Prediction and Evaluation: Make test outcome prediction using trained Ridge regression models (store predictions in y_pred), evaluate model accuracy, F1 score and display confusion matrix.

15. MODEL TESTING ON UNKNOWN SAMPLES

Query Sample Encoding

For new, unseen samples are pre-processed and encoded to high-dimensional vectors with the same N-gram encoding method. This allows for the new data to be compatible with what our models were trained on.

Prediction and Evaluation

On this annotated dataset, the models make predictions on the unknown samples and based upon their results we get to know how well do these models generalizes into new data via scores like Accuracy, F1 Score etc.

New Sample Prediction

The predicted language labels for new text samples are displayed, demonstrating the models' capabilities in real-world scenarios.

16. CONCLUSION

Overall, we saw that this project helps with the fact that n-gram encoding clearly works well and is highly accurate from our comparison results of Ridge Regression and Decision Tree classifiers. Results reveal that both models, given optimal training and evaluation protocols well exceed human-level accuracy at the task of language detection purely from textual data. Ridge Regression uses L2 regularization to serve the high-dimensional feature space, while Decision Trees provide a straightforward and non-linear method for classifying data.

This suggests that given a text n-gram encoding is able to easily capture the context and sequence dependency of words/characters in it and hence provides an efficacious way for machine learning models classifying language. In future work, advanced neural network architectures like recurrent neural networks (RNNs) or transformers can be exploited to enhance classification accuracy and efficiency. Furthermore, we could extend the dataset with extra languages and higher diversity in text genre to perform broader investigations on generalization across models.

17. REFERENCES

1. Scikit-learn Documentation: https://scikit-learn.org/stable/
2. Leipzig Corpora Collection: https://wortschatz.uni-leipzig.de/en/download
3. NLTK Documentation: https://www.nltk.org/