Project Report: Name Language Classification Using RNN

Raman 23MDT1048

1 Introduction

The aim of this project is to classify names by their language or region of origin using a Recurrent Neural Network (RNN). Language classification is a powerful tool in natural language processing (NLP) and language detection, aiding services such as personalized content recommendations, cultural analysis, and user profiling. This project leverages an RNN trained on a dataset of names labeled by language, enabling predictions on the language or regional origin of a given name.

2 Project Objectives

The primary objectives of this project are:

- To design and implement an RNN-based model for name language classification.
- To evaluate and improve the model's accuracy and robustness.
- To provide an interactive Streamlit app that allows users to test the model's predictions.

3 Data Preparation

3.1 Dataset Description

The dataset used in this project is a collection of names, each labeled with a language or region of origin. It includes names from several languages such as English, French, German, Spanish, Italian, Japanese, Chinese, and others, totaling 18 categories.

3.2 Data Preprocessing

Names are converted to a tensor of one-hot encoded vectors, representing each letter as a unique position in the vector. This encoding transforms the name

into a numerical format suitable for processing by the RNN. Here's a summary of the steps:

- **Character Encoding**: Each character in the name is represented as a one-hot vector, where each letter corresponds to a specific index.
- **Sequence Input**: Names are converted into sequences of character vectors, with each sequence fed into the model as an individual sample.

3.3 Dataset Statistics

- Number of categories (languages): 18
- Number of unique characters: 57 (including letters, punctuation marks, and spaces)

4 Model Architecture

4.1 Recurrent Neural Network (RNN)

The RNN architecture is well-suited for sequential data, enabling the model to learn dependencies between characters in names. The architecture consists of:

- **Input-to-Hidden Layer (i2h)**: Transforms the input character into the hidden state.
- **Hidden-to-Hidden Layer (h2h)**: Maintains the hidden state across time steps.
- **Hidden-to-Output Layer (h2o)**: Outputs the probability distribution over the language categories.
- **Activation**: A Tanh activation function is used for the hidden layer, while LogSoftmax is used in the output layer for classification.

4.2 Model Summary

- **Input Size**: 57 (size of one-hot encoded vectors)
- **Hidden Size**: 128 (number of hidden units)
- **Output Size**: 18 (number of language categories)

5 Training the Model

5.1 Training Process

The training process involves:

- **Forward Pass**: Each name is passed through the RNN to compute output probabilities for each language.
- **Loss Computation**: Cross-entropy loss compares the predicted label with the actual label.
- **Backpropagation and Optimization**: Loss is backpropagated, and weights are updated using the Adam optimizer.

5.2 Training Hyperparameters

- Learning Rate: 0.005
- Number of Epochs: 1000
- Batch Size: Each sample (name) is passed individually.

5.3 Challenges and Mitigations

- **Data Imbalance**: Some languages have fewer names, so data augmentation could be applied in future work.
- **Overfitting**: The model could overfit due to small sizes of certain language categories. Cross-validation and regularization techniques could help mitigate this.

6 Evaluation

6.1 Evaluation Metrics

- **Accuracy**: How often the model correctly predicts the language of names.
- **Confusion Matrix**: Shows prediction distribution across languages, identifying biases or common misclassifications.

6.2 Confusion Matrix Analysis

The confusion matrix shows the model's performance across languages, revealing that languages with similar linguistic characteristics (e.g., Romance languages like French, Spanish, Italian) are sometimes misclassified, suggesting areas for further training.

7 Streamlit Application

An interactive app was developed using Streamlit to provide a user-friendly interface for the classification model. Key features include:

- **Input Field**: Users enter a name for classification.

- **Prediction Output**: Displays the predicted language and confidence score.

To run the app, use:

```
streamlit run app.py
```

8 Results and Analysis

8.1 Model Performance

The RNN model achieved a satisfactory level of accuracy, improving as more training samples were processed. Performance may vary depending on the number and quality of names per category.

8.2 Insights

The model's misclassification rates indicate that some languages are closer in structure and character patterns, leading to occasional missclassifications.

9 Limitations and Future Work

9.1 Current Limitations

- **Data Imbalance**: Some language categories are underrepresented, affecting accuracy.
- **Language Similarities**: The model sometimes confuses languages that share similar naming patterns.

9.2 Future Enhancements

- **Data Augmentation**: Increase dataset size through augmentation or by sourcing additional names to balance categories.
- **Model Improvement**: Experiment with LSTM, GRU, or transformer architectures.

10 Conclusion

This project demonstrates the potential of RNNs for sequential data processing in classifying names by language. With future improvements, this project could become a valuable tool for language detection, user profiling, and cultural data analysis.

Appendices

- **Code Documentation**: Detailed comments and function documentation are available in the Python scripts.
- **References**: Techniques and implementations were inspired by resources on RNNs and PyTorch tutorials.