Title: Afaan Oromoo Next Word Prediction

Project Overview

Next word prediction is a fundamental task in natural language processing (NLP) that involves predicting the subsequent word in a given sequence of words.

This capability is crucial in various applications, including text generation, autocomplete systems and language translation. Our project leverages state-of-the-art machine learning models to develop a robust next word prediction system.

The aim is to facilitate better communication and support for speakers of this language by providing contextually accurate word suggestions.

The project faced significant challenges, including a critical memory error encountered during model training, specifically: `MemoryError: Unable to allocate 20.4 GiB for an array with shape (116586, 23445) and data type float64`. This document outlines the project's objectives, methodology, the encountered memory issue, and the strategies implemented to overcome it.

[W 22:57:24.653 NotebookApp] IOPub data rate exceeded.

The notebook server will temporarily stop sending output

to the client in order to avoid crashing it.

To change this limit, set the config variable

`--NotebookApp.iopub_data_rate_limit`.

Current values:

NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)

NotebookApp.rate_limit_window=3.0 (secs)

Objectives

The primary objectives of this project are:

1. To Develop an Accurate Afaan Oromo Next Word Prediction Model: Our aim is to create a model capable of predicting the next word in a sequence with high accuracy, enhancing user experience in text-based applications.

2. To Evaluate Different Model Architectures: We assess various NLP models, including Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks,

3. To Optimize for Real-Time Applications: Ensuring the model performs efficiently in real-time scenarios, suitable for applications like typing assistants or suggestion.

4. Address Data and Resource Constraints: Efficiently manage large datasets and computational resources to handle the unique challenges posed by the memory error.

Methodology

1. Data Collection and Preprocessing:

- We get dataset of Afaan Oromoo text sequences from mr.Tashome to train the model.

- Preprocessing involved tokenizing the text into words, converting words to numerical representations using techniques like word embeddings, and organizing data into sequences of a fixed length for model training.

2. Model Selection:

- LSTM Model: We implemented an LSTM-based model to leverage its ability to capture long-term dependencies in sequential data. LSTMs are particularly suited for handling the vanishing gradient problem in long sequences.

3. Training and Evaluation:

- Models were trained using the collected dataset, with training parameters fine-tuned to optimize performance.

- Evaluation metrics included accuracy, perplexity, and BLEU score to assess the quality of predictions and the model's ability to generalize to unseen data.

- We employed the `ModelCheckpoint` callback to save the best-performing model based on the monitored metric (`loss`), ensuring optimal model retention during training.

```
from tensorflow.keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint("next_words_h5.keras", monitor='loss', verbose=1, save_best_only=True)
```

4. Optimization for Real-Time Usage:

- We explored smaller or distilled versions of models to balance prediction quality and computational efficiency.

- Techniques like quantization and pruning were considered to reduce model size and improve inference speed without significant loss in accuracy.

5. Memory Management and Error Handling:

- Encountered MemoryError: The error occurred when trying to allocate a large array during the data processing or model training phase. This required a re-evaluation of the data handling and model architecture.

Solutions Implemented:

- Data Reduction: Reduced the dataset size and used more efficient data types to lower memory usage.

- Model Optimization: Reduced the model size and complexity, and explored using lower-precision data types (like float32 instead of float64) to cut down memory requirements.

Implementation Details

LSTM Model Implementation:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Embedding


# Define the LSTM model
model= Sequential()
model.add(Embedding(total_words,100, input_length= max_sequence_length - 1))
model.add(LSTM(150))
model.add(Dense(total_words,activation='softmax'))
# Compile the model
```

```
  model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])


  # Train the model
  model.fit(x_train, y_train, epochs=100, batch_size=64,
validation_data=(x_val, y_val))
```

R**esults ,Evaluation and Future Work**

Our experiments demonstrated that:

- The LSTM model performed well on tasks with shorter contexts and simpler vocabulary, achieving high accuracy and reasonable perplexity.

Moving forward, we plan to:

- Fine-tune Models on Specific Domains: Tailor models for particular applications, such as medical text or legal documents, to improve domain-specific accuracy.

- Explore More Efficient Model Architectures: Investigate lightweight architectures and model compression techniques to enhance performance in resource-constrained environments.

- Enhance User Interaction: Develop interactive applications that leverage the next word prediction model for dynamic text generation and assistance.

- Enhancing Model Efficiency: We will explore further optimizations, including model pruning and quantization, to enhance performance without sacrificing accuracy.

- Expanding Dataset: As more data becomes available, incremental training methods will be adopted to expand the model's vocabulary and context understanding while managing memory effectively

Conclusion

The project successfully developed a Afaan Oromoo next word prediction. Overcoming significant memory challenges. The experience gained and solutions implemented provide a solid foundation for future advancements in integrating local languages into technological applications.

Project Acknowledgments

This project would not have been possible without the support and resources provided by several key sources and individuals

1. Teacher's Contribution:

I extend my deepest gratitude to my teacher, mr.teshome for their significant contribution to this project. They not only provided the essential dataset but also offered continuous encouragement and guidance throughout the development process.

2. YouTube Video:

Python Machine Learning Tutorial for Beginners.

URL: https://www.youtube.com/watch?v=RnFGwxJwx-0

3. AI Assistance:

I received significant assistance from OpenAI's GPT-4 tool, ChatGPT, which provided detailed guidance on coding and implementation strategies https://www.openai.com/chatgpt

URL: https://www.openai.com/chatgpt

Each of these resources and individuals has contributed significantly to the successful completion of this project.