**Group 8: Final Project**

**Sanskrit-English Language Translator using Bhagavad Gita Parallel Corpus.**

* **Group members:**

Atharv Amit Kadam

Aditya Parimal Soni

Nikhil Vashisht

Nirali Nilesh Mody

Parthkumar Hiteshbhai Gohil

* **Project Overview:**

The Sanskrit-English Language Translator project aims to bridge the linguistic gap between Sanskrit and English, with a primary focus on translating sacred texts, specifically the Bhagavad Gita. Leveraging a parallel corpus consisting of the Bhagavad Gita in both Sanskrit and English, we have developed a powerful LSTM (Long Short-Term Memory) neural network model capable of accurately translating between these two languages.

* **Key Features:**
  1. Parallel Corpus Integration: The project utilizes a carefully curated parallel corpus containing the Bhagavad Gita in Sanskrit and its corresponding English translation. This corpus serves as the foundation for training the language translation model.
  2. Text Preprocessing: The Sanskrit and English texts undergo preprocessing steps, including removing punctuation, lowercasing English text, and tokenization. This ensures that the input data is clean and suitable for training the neural network.
  3. Word Mapping and Vocabulary Analysis**:** The project analyzes the unique words and vocabulary size in both Sanskrit and English. The word mapping process aligns Sanskrit and English words, contributing to the neural network's understanding of semantic relationships.
  4. LSTM Neural Network: The heart of the translation system is an LSTM neural network, a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data. The architecture enables the model to understand the contextual nuances present in the Bhagavad Gita and generate accurate translations.
  5. Training Data Preprocessing: Rigorous preprocessing of the training data involves tokenization and padding sequences to ensure uniform input lengths, facilitating effective training of the LSTM model.
  6. Embedding Model: The project incorporates an embedding model, utilizing Keras and TensorFlow, to represent the input sequences in a way that the neural network can understand. This step involves assigning numerical values to words and training the model for accurate translations.
  7. Text-to-Speech (TTS): The project includes functionality for both English and Sanskrit Text-to-Speech conversion. This feature allows users to listen to the translated text, enhancing the accessibility and usability of the translation system.
* **Applications:** 
  1. Religious Studies: Scholars and practitioners can utilize the translator to gain a deeper understanding of the Bhagavad Gita's teachings by exploring the nuances in both Sanskrit and English.
  2. Language Learning: The project serves as a valuable resource for language learners interested in Sanskrit or English, offering an opportunity to study classical languages in the context of a revered spiritual text.
  3. Cross-Cultural Communication: Facilitates communication and understanding between speakers of Sanskrit and English, fostering cross-cultural exchange.
* **Future Developments:**
  1. Continuous Refinement: The model can be further refined with additional training data and improved preprocessing techniques to enhance translation accuracy.
  2. User Feedback Integration: User feedback can be valuable for refining the model's performance and addressing specific translation challenges.
  3. Multilingual Expansion: The translator's capabilities can be expanded to include other classical languages and sacred texts, broadening its applicability and cultural impact.
* **Brief Code Explanation:**
  1. **Data Reading and Preprocessing:** Reading Bhagavad Gita texts in Sanskrit and English, removing unnecessary characters, and conducting initial text preprocessing.
  2. **Vocabulary Analysis:** Analyzing the unique words and vocabulary size in both Sanskrit and English.
  3. **Tokenization:** Using Keras Tokenizer to convert text data into sequences of numerical values.
  4. **Padding Sequences:** Ensuring uniform sequence lengths for efficient model training.
  5. **Embedding Model Creation:** Creating an LSTM embedding model using Keras, TensorFlow, and Adam optimizer for training.
  6. **Model Training:** Training the LSTM model with the prepared data and saving the model for future use.
  7. **Text-to-Speech Conversion:** Implementing Text-to-Speech functionality for both English and Sanskrit translations.
  8. **Limiting Translated Text:** Reducing the translated text output to avoid repetition.
  9. **Text-to-Speech Execution:** Demonstrating the project's capability by converting translated text into audio files for both English and Sanskrit.
* **LSTM embedding model:**

**A screenshot of a computer program

Description automatically generated**

* The provided code outlines the construction of a neural network model geared for sequence-to-sequence translation, notably applied to the task of translating Sanskrit to English.
* The model is encapsulated within a function named embed\_model. It starts by initializing a sequential model, a linear stack of layers.
* The architecture unfolds with the incorporation of an Embedding layer, responsible for converting integer-encoded words into dense vectors of fixed size (128 dimensions), where the length of the input sequences is determined by the output\_sequence\_length parameter.
* Subsequently, an LSTM (Long Short-Term Memory) layer, equipped with 128 units, is introduced to capture sequential dependencies, and maintain context across the input sequences.
* A Dropout mechanism with a rate of 0.1 is applied to mitigate overfitting, and the return\_sequences=True setting ensures the production of sequence outputs.
* The model culminates with a Dense layer, comprising units equal to the size of the English vocabulary, and utilizes a softmax activation function to yield output probabilities for each word in the target vocabulary.
* The summary of the model's architecture, showcasing layer details and parameter counts, is printed for reference. Following the architectural definition, the model is compiled for training, specifying the sparse categorical crossentropy loss function, the Adam optimizer with a customizable learning rate (defaulted to 0.1), and accuracy as the evaluation metric. The completed model is then returned, ready for further training and utilization in the Sanskrit-English translation endeavor.
* There are several activation functions used in neural networks, each serving different purposes. Some of the commonly used activation functions include:

**Sigmoid Activation Function:** Outputs values between 0 and 1, often used in binary classification problems.

**Hyperbolic Tangent (tanh) Activation Function:** Similar to the sigmoid but outputs values between -1 and 1, often used in the hidden layers of neural networks.

**Rectified Linear Unit (ReLU) Activation Function:** Outputs the input for positive values and zero for negative values. It is widely used in hidden layers due to its simplicity and effectiveness.

**Leaky ReLU Activation Function:** Similar to ReLU but allows a small, positive gradient for negative values, addressing the "dying ReLU" problem.

**Exponential Linear Unit (ELU) Activation Function:** A variant of ReLU with smoother outputs for negative values.

**Softmax Activation Function:** Used in the output layer for multi-class classification problems. It converts the raw output scores into probabilities, ensuring that the sum of the probabilities across all classes is equal to 1.

* Here we used softmax activation function in the output layer. This choice is appropriate for sequence-to-sequence translation tasks where each word in the target language (English, in this case) needs to be assigned a probability. The softmax function normalizes the output scores into a probability distribution, facilitating the selection of the most probable word during training and inference. Specifically, in machine translation, it ensures that the model produces valid probability distributions over the entire target vocabulary, aiding in the generation of coherent and meaningful translations.
* **Text-to-Speech (TTS):**
* **English Text-to-Speech (enTTS):**
  + - Initialization: The pyttsx3 library is utilized for English TTS. The engine is initialized, voices are obtained, and a specific voice (index 1) is set.
    - Speech Rate Setting: The speech rate is configured to 150 words per minute (rate = 150).
    - Text-to-Speech Conversion: The engine is instructed to say "English" and the text obtained from the limiten function (which limits the translated output until a word repeats).
    - Saving to File: The generated speech is saved to an MP3 file using the save\_to\_file method, and the file is named based on the input index.
    - Execution and Confirmation: The TTS engine is run, and a confirmation message is printed indicating the successful saving of the MP3 file.
* **Sanskrit Text-to-Speech (saTTS):**
  + - Initialization: The gTTS library is employed for Sanskrit TTS. The Sanskrit text corresponding to the given index n is extracted from array ‘x’.
    - Text-to-Speech Conversion: A gTTS object is created with the Sanskrit text, specifying the language as 'hi' (Hindi/Sanskrit) and setting slow to False for normal speech speed.
    - Saving to File and Playback: The generated audio is saved as an MP3 file named based on the input index. Subsequently, the pygame library is used to play the generated audio.
    - Confirmation: A confirmation message is printed once the audio file is saved.
* Top of Form
* **Project Collaboration Breakdown:**
  1. Atharv Amit Kadam: Data Preparation and Initial Preprocessing
  2. Nikhil Vashisht: Vocabulary Analysis and Tokenization
  3. Nirali Mody: Length Analysis, Padding, Most Common Words
  4. Aditya Parimal Soni: LSTM Embedding Model
  5. Parthkumar Hiteshbhai Gohil: Text to Speech
* **Conclusion:**
* In conclusion, the language translation project seamlessly integrates advanced natural language processing technique, deep learning model, and text-to-speech capabilities to bridge linguistic gaps between Sanskrit and English. Leveraging a meticulously trained LSTM neural network model with a commendable accuracy of 74%, the project not only facilitates accurate translation but also extends the user experience through synthesized speech output. The code showcases a holistic approach, encompassing data preprocessing, model training, and the practical application of text-to-speech functionality for both languages.
* **Future advancement:**
* The future advancement potential of the project is vast, primarily by exploring the adaptation of the language translation model to diverse language datasets. By extending the training data to include additional language pairs, the system can evolve into a versatile multilingual translator. This expansion could involve incorporating datasets for languages spanning various linguistic families and geographical regions, thereby enhancing the model's ability to comprehend and translate a broader spectrum of languages.