|  |  |
| --- | --- |
|  |  |
|  |
|  |

**High Impact Skills Development Program for Gilgit Baltistan**

**Natural Language Processing Project**

**Design and Development of Topical Chatbot**

**Submitted By: Rukhsana**

**GIL-DSAI-095**

https://github.com/RukhsanaAyub

**Introduction:**

A topical chatbot is a specialized AI designed for conversations about a specific subject. Unlike general chatbots, these focus on one topic, like medicine or finance. They're trained to understand and discuss that area, using datasets of relevant conversations. Topical chatbots provide accurate and helpful information within their domain, enhancing user engagement and knowledge in that subject. However, they may struggle with topics beyond their training and lack broader conversational abilities.

**Dataset:**

The dataset used for this project is Topical Chat dataset from Amazon. It consists of over 8000 conversations and over 184000 messages. Within each message, there is: A conversation id, which is basically which conversation the message takes place in. Each message is either the start of a conversation or a reply from the previous message. There is also a sentiment, which represents the emotion that the person who sent the message is feeling. There are 8 sentiments: Angry, Curious to Dive Deeper, Disguised, Fearful, Happy, Sad, and Surprised.

**Architecture:**

Creating a chatbot using TensorFlow and a Transformer model involves building a sequence-to-sequence model, where the encoder encodes the user input, and the decoder generates the chatbot's response.

Here's a simplified version of a chatbot implemented using TensorFlow, which leverages the Transformer architecture. This implementation will focus on the following:

1. **Preprocessing the data**

a. Tokenization: Splitting sentences into individual words or sub-words.

b. Padding: Making sure all input and output sequences are of uniform length.

c. Vectorization: Converting text into numerical format (e.g., using word embeddings or one-hot encoding).

d. Data Augmentation: Generating more training data through paraphrasing or synonym replacement.

2. **Building the model** – Using a simplified Transformer architecture with an encoder and a decoder.

#### Encoder-Decoder Model

For a chatbot, we generally use a sequence-to-sequence (Seq2Seq) architecture, which uses an encoder-decoder framework. The encoder processes the input sentence, and the decoder generates the response.

The Transformer model extends this basic architecture with:

* Multi-Head Attention: Allows the model to attend to different parts of the input sequence simultaneously, improving context understanding.
* Positional Encoding: Since Transformers do not process data sequentially, positional encoding are added to the input to retain information about the order of the words.
* Feed-Forward Networks: Added after the attention mechanism for further processing of information.

3. **Training the model**

#### Loss Function

For chatbot generation, the most commonly used loss function is categorical cross-entropy, where the model learns to predict the next word in the sequence.

#### Optimization

During training, the model parameters are updated using optimizers like Adam or RMSProp. These optimizers adjust the model weights based on the gradients of the loss function.

#### Hyper parameters

Key hyper parameters include:

* Learning Rate: Controls how much to update the model during each training step.
* Batch Size: The number of training examples used in one update.
* Number of Layers: The number of encoder-decoder layers.
* Hidden Units: The size of the hidden layers in the model.

#### Training Process

Training typically involves the following steps:

1. Initialize Model: Define the Transformer architecture in TensorFlow.
2. Feed Training Data: Provide pairs of input-output sequences.
3. Optimization: Use an optimizer to minimize the loss function.
4. Evaluation: Periodically evaluate the model's performance on a validation set.

4. **Generating responses** – The chatbot generate responses based on the trained model. Once trained, we can use the `generate\_response` function to take user input, convert it to a sequence, and generate a response based on the model's prediction.

Once the chatbot is trained and evaluated, it can be deployed in an application, such as:

* Web Applications: Using frameworks like Flask or Django to integrate the chatbot into a website.
* Messaging Platforms: Platforms like Slack, Facebook Messenger, or WhatsApp can be integrated with a chatbot API.

For real-time applications, the chatbot should be optimized for speed and efficiency. Techniques like model quantization and pruning can be used to reduce the model's size and improve inference time.