**Generative Chatbots**

Application of Neural Networks and Deep Learning

Project Synopsis

Project Mentor

Dr. Asoke Nath

Project Members

Shreyashi Choudhary – 502

Arighna Chakraborty – 511

Tiyasa Chatterjee – 521

**Abstract**

Dialogue generation or development of intelligent conversational chatbots using Artificial Intelligence or Machine Learning techniques is an interesting problem in the field of Natural Language Processing. Conversational chatbots are predominantly used by businesses, government organizations and financial organizations. There are many frameworks and existing chatbots available, but they generally lack the flexibility in developing real-world conversations. The functionality of these bots are limited since most of them are retrieval based (are used in closed domain scenarios and rely on a collection of predefined responses), and also they are not aimed at holding conversations that emulate real human interaction. **[1]**

This project aims to develop and implement a simple *Generative Chatbot* using *Transformer* models **[2]**.*Generative chatbots* are *open domain* systems that generate original combinations of language rather than selecting responses from predefined data. The new combinations of responses are generated using various machine learning and deep learning techniques feeding upon a lot of historic data and previous conversations. The *Transformer* model network architecture , based solely on attention mechanisms, dispensing with recurrence and convolutions entirely **[8]**. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train.

**Introduction**

Chatbot applications streamline interactions between people and services, enhancing customer experience. At the same time, they offer companies new opportunities to improve the customer’s engagement process and operational efficiency by reducing the typical cost of customer service.

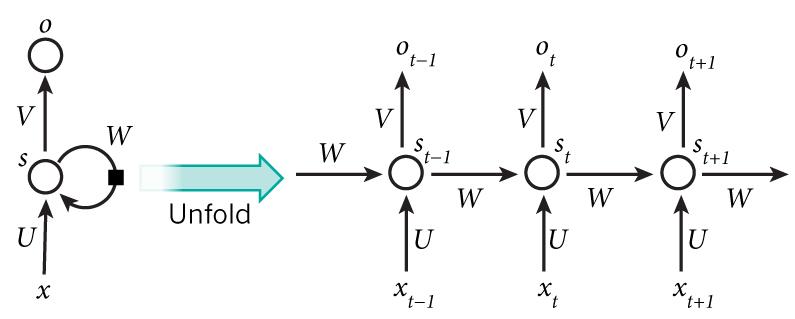
Generative chatbot research is currently working to resolve how best to handle chat context and information from previous turns of dialogue. Although the process does not rely on predefined responses, the responses given might be irrelevant or grammatically incorrect in many cases. This project deals with how such a conversational bot can be implemented using Transformer and Sequence to Sequence models.

The core idea behind the Transformer model is *self-attention* **[9]**—the ability to attend to different positions of the input sequence to compute a representation of that sequence. Transformer creates stacks of self-attention layers. **Note:** The model architecture is identical to the example in the *Transformer model for Language Understanding* **[2]**, and we demonstrate how to implement the same model in a Functional approach.

**Underlying Concepts**

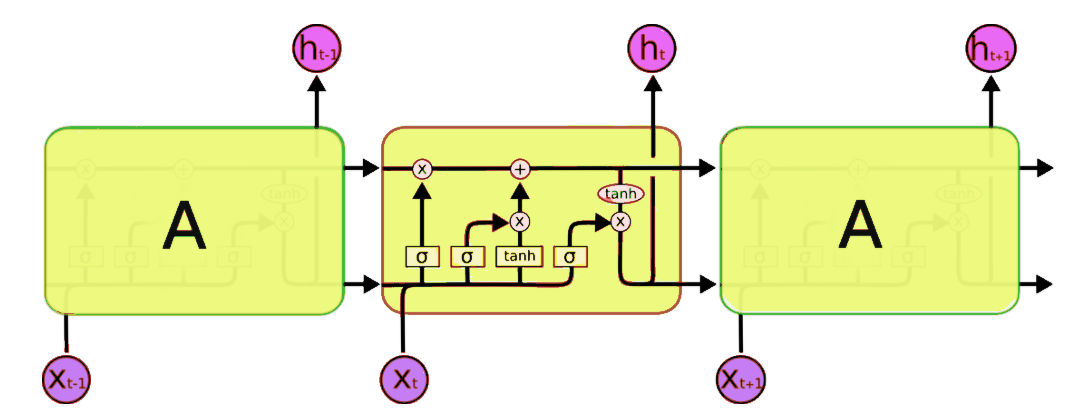
* **Recurrent Neural Networks**

Recurrent Neural Networks **[3]** are powerful models that are uniquely capable of dealing with sequential data, like natural language, speech synthesis etc. They take a fixed-sized vector as input and produce a vector output. RNNs are called recurrent because they perform the same task for every element of a sequence **[4]**, with the output being dependent on the previous computations.

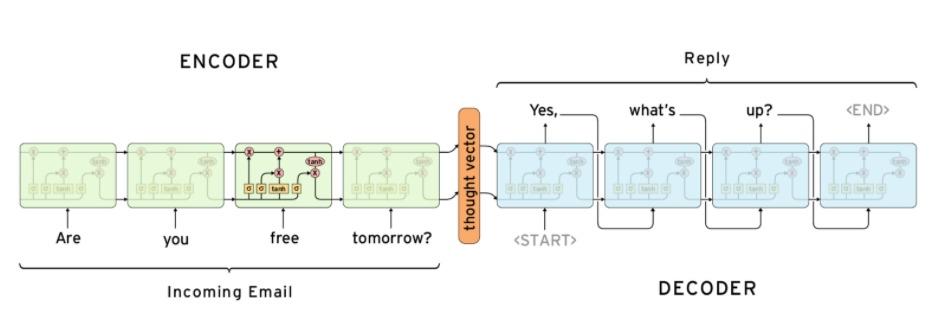


* **Long Short Term Memory Networks**

Long Short Term Memory networks **[5]** – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem in traditional RNNs. Remembering information for long periods of time is practically their default behaviour.

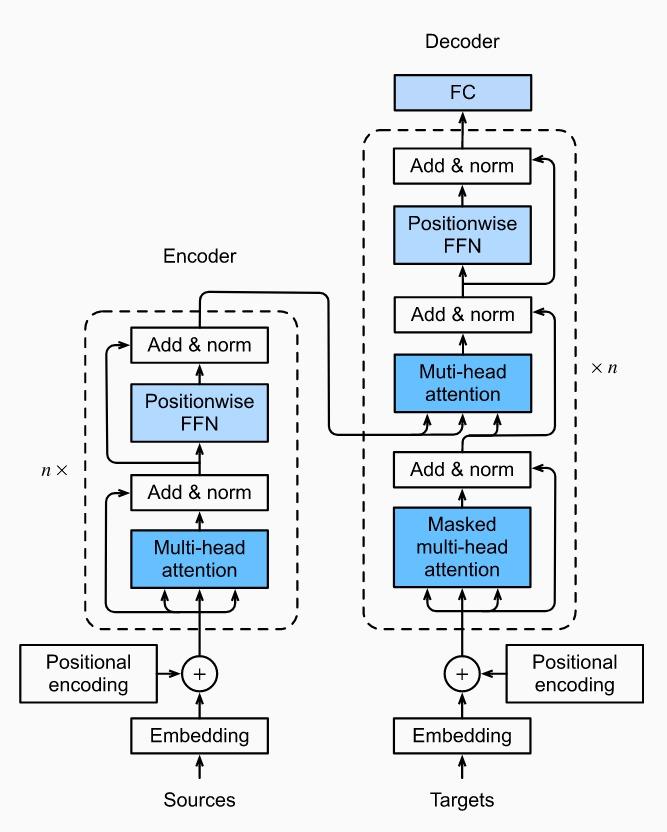


* **Sequence to Sequence Models**

Sequence To Sequence model **[6]** in recent times has become the go-to model for Dialogue Systems and Machine Translation. It consists of two RNNs (more specifically two LSTMs) - An Encoder and a Decoder **[7]**. The encoder takes a sequence (sentence) as input and processes one symbol (word) at each timestep. The output from the encoder is called the context or thought vector, as it represents the intention of the sequence. From the context, the decoder generates another sequence, one symbol (word) at a time.

* **Transformer Model**

Most sequence generation models have an encoder-decoder structure ( as in sequence to sequence model ). The transformer model **[8]** follows the same overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. Self-attention **[9]**, sometimes called intra-attention, is an attention mechanism relating different positions of a single sequence to compute a representation of the same sequence. The outputs are aggregates of these interactions and attention scores.



**Proposed Procedure**

All the theories aside, the actual implementation of a *Transformer* model for an open domain generative chatbot involves a lot of intermediate steps and processing. Here are the overviews of the steps involved in the implementation of such a model,

1. **Selection of a dataset**

An effective chatbot requires a massive amount of training data to quickly and efficiently reply to user questions. However, the main obstacle to the development of a chatbot is obtaining realistic and task-oriented dialogue data to train these machine learning-based systems. *A deep learning model is only as good as the training data fed into it.* As the chatbot not only answers user questions but also converses with them, it becomes imperative that the correct dataset is used for training the model. We are using corpus from *Twitter* (700K lines), where the odd lines are tweets and the even lines are corresponding responded tweets as well as we are also using the *Cornell movie* dataset which contains a large collection of fictional conversations extracted from raw movie scripts.

*The datasets used can be found in zipped format here*: [*Chat Corpus*](https://github.com/Marsan-Ma/chat_corpus/)

1. **Preprocessing the dataset**

The training process of such complicated deep learning models is very resource-intensive. To keep the training simple and fast, the datasets need to be further processed before feeding them to the neural network model.

Processing of the dataset further includes,

* Extract conversation pairs into a list of questions and answers.
* Removing special characters and abbreviated texts (I’m, he’s, you’re and so on).
* *Tokenization* - mapping words to IDs and IDs to words (each word is a token) and adding START\_TOKEN and END\_TOKEN to indicate the start and end of each sentence.
* Setting a MAX\_LENGTH for questions and filter out sentences that have more than MAX\_LENGTH tokens.
* Padding the tokenized sentences to MAX\_LENGTH.

1. **Training the Model using processed data**

We are planning on incorporating the Tensorflow Dataset API ([tf.data.Dataset](https://www.tensorflow.org/api_docs/python/tf/data/Dataset)) to construct our input pipeline to utilize features like caching and prefetching to speed up the training process (only a concept at this point). The transformer is an auto-regressive model: it makes predictions one part at a time and uses its output so far to decide what to do next.

As the *Transformer* predicts each word, *Self-Attention* allows it to look at the previous words in the input sequence to better predict the next word.

1. **Testing the Model**

After successful training of the model on a fairly large number of epochs (iterations), and an appropriate learning rate ( 0.03 <= LR <= 0.3 ), the loss function (also called cost function) of the model converges at the global minima.

Now, the model is ready to be tested with real-world conversations.

**Conclusion and Future Scope**

This is a largely simplified report describing the planning and the overall synopsis of our project. The actual concepts, workings and results may deviate depending on the dataset, processing of the data, tweaking the numerous parameters and the challenges faced during the actual implementation and training of the model.

The future scope of this project involves:

* Gathering and preparing more diverse datasets both in data quality and in size.
* Further tweaking and improving the neural network structure and parameters for an increase in efficiency and accuracy of performance and results.
* Implementing text-to-speech features for a more interactive experience.
* Upgrade the model to implement a Hybrid Chatbot that utilises both Retrieval and Generative dialogue generation.
* Adding a memory system to implement continued conversations (where the chatbot remembers the previous conversations).
* Abstracting the model behind a GUI.

Deploying a fully functional open domain generative chatbot from scratch is a fairly complicated endeavour considering the lack of hardware resources and computing power. For example, the state-of-the-art *GPT-3* model from Open AI uses 1.75 billion parameters and a dataset of about 100 gigabytes. Achieving the accuracy and efficiency even remotely close to GPT-3 requires a high-performance system which is certainly out of our scope.

However, there have been attempts to re-create the exceptional accuracy and performance of the GPT-3 model. *GPT-neo* is one such model which is *“an implementation of model and data-parallel GPT-3 like models using the mesh tensorflow library”* **[10]**. Another project is currently in the works – *GPT-neox* to further optimize the training process by allowing developers to scale model training to hundreds of billions of parameters across multiple GPUs.

**References**

**[1]** *Asoke Nath, Rupamita Sarkar, Swastik Mitra, Rohitaswa Pradhan*, **“Designing and Implementing Conversational Intelligent Chatbot Using Natural Language Processing”**, *International Journal of Scientific in Computer Science, Engineering and Information Technology (IJSRCSEIT)*, 22nd May 2021.

**[2]** [**Transformer model for language understanding**](https://www.tensorflow.org/text/tutorials/transformer)**,** *Tensorflow Documentation*

**[3]** *Denny Britz*, **“Recurrent Neural Network Series”**, *Wild ML,* 17th September 2015

**[4]***Toma´s Mikolov, Stefan Kombrink, Luka´s Burget, Jan “Honza” Cernock, Sanjeev Khudanpur,* **“Extensions of Recurrent Neural Networks Language Model”**, *John Hopkins University USA*, 1st June 2011

**[5]** *Christopher Olah*, **“Understanding LSTM Networks”**, Colah’s Blog, 27th August 2015

**[6]** *Ilya Sutskever, Oriol Vinyals, Quoc V.Le,* **“Sequence to Sequence Learning with Neural Networks”**,[*arXiv:1409.3215*](https://arxiv.org/abs/1409.3215)*,* 10th September 2014

**[7]***Kyunghyun Cho, Bart vanMerrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio,* **“Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”**, [*arXiv:1406.1078*](https://arxiv.org/abs/1406.1078)**,** 3rd June 2014

**[8]***Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin,* **“Attention is All You Need”,** [*arXiv:1706.03762*](https://arxiv.org/abs/1706.03762), 6th Dec 2017

**[9]** *Buomsoo Kim*, **“Attention in Neural Networks”,** 11th November 2020

**[10]** *EleutherAI,* **GPT-neo and GPT-neox,** [*EleutherAI (github.com)*](https://github.com/EleutherAI)