Chatbot Analytics Based on Question Answering System Movie Related Chatbot Case Analytics

Jugal Shah [1] and Sabah Mohammed [2]

[1]Lakehead University
[2]Affiliation not available

October 30, 2023

Abstract

In this paper, an attempt has been made to understand the importance of a neural network-based chatbot system for movie-related queries.

Chatbot Analytics Based on Question Answering System: Movie Related Chatbot Case Analytics

Jugal Shah
Department of Computer Science
Lakehead University
Thunder Bay, Canada
jshah5@lakeheadu.ca

Dr. Sabah Mohammed
Faculty of Computer Science
Lakehead University
Thunder Bay, Canada
mohammed@lakeheadu.ca

***Abstract*—Question Answer (QA) systems are established to retrieves accurate and concise answers to human queries posted in natural language. The primary focus of the QA system is to achieve efficient and natural interaction between machines and humans. To achieve the above several researchers are directed towards Natural Language Processing (NLP) based deep learning. With the rise of a variety of deep NLP models, it is now possible to obtain a vector form of words and sentences that stores the meaning of the context. NLP considerably aids deep learning-based mathematical models in understanding the semantic and syntax of natural human language. In this paper, research is conducted on chatbots based QA system. The sequence-to-sequence (seq2seq) model proposed by Ilya Sutskever 2014, has laid the foundation for the chatbot model build in this paper. The Cornell Movie-Dialogs Corpus created at Cornell University, and Movie Dialog Dataset created at Facebook are preprocessed and used to train the chatbot. The encoder and decoder of the seq2seq model comprise of LSTM cells and are defined using Bidirectional Dynamic RNN and Dynamic Decoder RNN package of the tensor flow library. Additionally, to ensure the chatbot performs well on long sentences attention mechanism from the tensor flow library is applied to the decoder.**

***Index Terms*—QA, LSTM, RNN, Seq2Seq, decoder, encoder, attention**

I. Introduction

Existing information retrieval systems retrieves a list of documents in response to queries inquired by a human in natural language. One example that is part of our day to day life would be searching on Google search. The list of documents presented to the user as results might have the correlated data; however, a considerable amount of work is done by humans to extract valuable information from the list of documents [1][2]. Hence with the need to get a concise and exact answer to user queries, Question Answer Systems came to existence. The QA system derives the answers from a variety of sources: unstructured data (e.g., Web pages, blogs), semi-structured data (e.g., Wikipedia), structured Knowledge-base [3]. Based on the domain of question answered by the QA system can be classified as open-domain QA or closed-domain QA. Closed- domain QA answer question to a specific field, whereas an open domain QA gives the user the freedom to raise queries related to any topic [4]. The QA system proposed for the travel domain in [5] can be a suitable example for closed-domain QA, whereas the QA system developed for factoid-based question [what, when, who, which, how] is an open-domain QA [6]. One of the metrics for the evaluation of this QA system is the accuracy of the answer. One way to reach the desired answer is by classifying the question into different categories [7]. The recent advancements in the field of NLP and deep learning, has led to the application of neural networks like RNN, BiLSTM in building an open-domain QA system [8] The neural networks are trained on natural human language, mostly by word embedding. Today, QA is an influential discipline of more sophisticated natural language processing (NLP) techniques [9]. The QA systems are categorized into four categories, chat robot, QA based knowledge base, QA retrieval system, and QA based on free text [10]. This paper concentrates on a chat robot-based QA system, also known as a chatbot.

II. Critical Review

Research has been conducted to address the problem of human-machine interaction for a very long period. The earliest chatbot ever build was developed by MIT called ELIZA, and it dates back to the 1960s. ELIZA follows a rule-based design methodology and operates on pattern matching and substitution algorithms. It gives an illusion of a program with an understanding [11]. ELIZA was build to offer Rogerian psychotherapy to patients by asking personal questions and engaging in a long conversation [12]. The bot function by analyzing the patient's response and performing keyword matching on the set of predefined templates to generate a formatted string response. The bot demonstrated the capability to provide aid to the patients; however, its efficiency could not match that of a human therapist. Additionally, ELIZA significantly lagged the ability to learn new patterns from interaction and perform logical reasoning due to the rule-based approach.

Today with the recent innovations in the field of machine learning and NLP, cloud-based chatbot platforms are generated. Various commercial tools like IBM Watson Conversation service has been established to assist in developing a chatbot. Neha Godse et al., 2018, showcased the application of cloud-based chatbot platform IBM Watson Conversation service for Information technology service management (ITSM) application in software companies [13]. The author proposes a chatbot that can help an employee within the company to resolve his or her IT-related issues without raising a ticket. The chatbot takes user input in natural language, bounds it into a JSON object, and sends it to IBM Watson using plugins. At IBM Watson Intents, Entities and Dialog flow have been defined for generating the response. Intents specify the purpose of the user query; entities specify the context of the intents, and dialog flow represents the conversation nodes that are created based on the intent and entities. The conversation node's response is then passed on to the user. Heru Santoso et al. 2018, also present the application of another cloud-based chatbot platform called DialogFlow for university admission services [14]. The author proposes a chatbot called DINA, Dinus intelligent assistant, that responds to student's university admission queries. Here the methodology mentioned by the above authors can be widely used when building a chatbot for close-domain like hotel reservations in which an expert can manually define the intents, entities, and dialog. However, it might not be an optimal solution for open-domain chatbots as the conversation scope is too broad. Additionally, the chatbot learning might be limited.

With messaging service being ubiquitous and the elevation of deep Learning and NLP techniques, scalable and critical learning became achievable, which contributed to the emergence of intelligent chatbot, an instance of which is A Neural Conversational Model [15][17]. These chatbots are not dependent on a predefined knowledge base or rules. They are deep learning models trained on conversation samples giving them the ability to respond to open-domain queries. The chatbot may belong to the generative model or retrieval-based model category [18]. The Automated Thai-FAQ Chatbot using RNN-LSTM proposed by Panitan et al., 2018, signifies a Retrieval based model [19]. In a retrieval-based model, the model learns to select a suitable response from a set of predefined responses, whereas the generative model generates a new response from scratch. A generative model trained on a vast corpus can generate better response compared to the retrieval-based model. Neural Conversational Model outlines a Generative Model. It involves the training of a neural network with extensive data to build a conversational model that can converse in natural human language. Sequence to Sequence framework proposed by Ilya Sutskever, 2014 laid the path to the Neural Conversational Model [16] [17]. The initial application of this framework involved neural machine translation and archival [20][17]. Huyen Nguyen et al. 2017, represents an instance of a neural chatbot based on the seq2seq model [22]. The authors propose an open-domain response generator that imitates characters from popular tv shows. The model as trained using five different datasets and was evaluated using automatic metrics BLEU and ROUGE as well as human judgment. Based on the parameters presented in the paper, the bot is able to communicate fluently, proving the capability of a seq2seq. Taking advantage of the seq2seq model, in this paper, we focus on the application of the seq2seq model to build a close-domain chatbot for movie-related questions and answers.

III. Problem Definition

Traditional chatbots are deeply dependent on hand-written rules. The chatbots architecture included a fixed template and some NLP based analytical method to generate the response. Also, these chatbots respond to questions limited to a specific domain. With the recent advances made in the field of Neural Network and NLP, in this paper, we experiment with a chatbot based on the Neural Conversation model suggested in [17]. A Neural Conversation model can be trained end-to-end, eliminating the bottleneck caused due to predefined hand-written rules. It is based on the Seq2Seq framework, which involves the use of LSTM. A well-known application of the Seq2Seq framework in the field of Machine translation.

In recent years researchers have also started applying the seq2seq framework for building a deep NLP based chatbot; most of this study is performed on an open-domain dataset that enables the chatbot to converse with the human in natural language. In this paper, the focus is to utilize the power of a seq2seq model to build a chatbot that can respond to close domain questions related to movies. For this study, the Cornell Movie-Dialogs Corpus created at Cornell University, and Movie Dialog Dataset created at Facebook would be used to train the chatbot.

IV. Methodology

In this research, we would be creating a prototype of a seq2seq model-based chatbot by first training the model on Cornell Movie-Dialogs Corpus. This corpus would help the chatbot to build the general conversation ability of the model. In the second stage, the model would be trained on the close-domain Movie Dialog Dataset, which would enable the chatbot to respond to queries specific to film. Below steps are performed to build the seq2seq model-based chatbot.

A. *Pre-processing*

The preprocessing of the Cornell and Facebook movie dataset is divided into multiple sections.

a) Data Harmonization: Considering both datasets Cornell Movie-Dialogs Corpus and Facebook Movie Dialog dataset exist in a different format; the initial step would be to convert both the dataset into a single unified form. To achieve a standard format of the data representation, we would represent all the data in question and response format. The question and answer list would have one to one mapping. This one to one mapping would assist in data preprocessing steps and would also simplify the model training. The above is achieved using basic python functionality.

b) Data Cleaning: As a part of data cleaning, the question and answers generated in the previous step are first converted to the lower case, this reduces the size of word vocabulary, making it easier and efficient for the model to learn. The lower case question and answers are then analyzed to expand the short words and remove the punctuation's from the text. The list of short words to be expanded and the punctuation to be removed is currently a static list that can be modified to add new conditions. For example, if the text is

”That’s all I’ve to say,” the cleaned version would be ”that is all i have to say’”.

c) Word Vocabulary: To generate the word vocabulary following steps are performed.

- Word Tokenization: In this step, the questions and answers are broken down into a list of words. Word tokenization can be achieved using the word_tokenize function of the python NLTK library or using Python elementary functions. For example, the text ”I told you, this is not you” would be tokenized to the list [i,told, you, this, is, not, you].
- Word occurrence count: At this step, we generate a dictionary of words in the corpus associated with their count. Word count can be performed using python FreqDist library or by iterating over the whole dataset.
- Frequent Words selection: The dictionary generated in the previous step is used to filter out words that occur less time than the given threshold. It is essential to limit the words because passing too much data to the model may affect the model training.
- Mapping Frequent words: All the frequent words identified are mapped to a unique integer.
- Tokens: The <SOS> and <EOS> tokens are used to indicate the start of the sentence and end of the sentence for a seq2seq model. Additionally, <PAD> token is to used to pad the question and answers so that all the questions are of the same length, and the <OUT> token is used to represent the non-frequent words in the sentence. Each token is assigned a unique number.

d) Dataset Vocabulary: The word vocabulary generated in the last step is used to represent each word in the question and answer list as an integer. Since only frequent words are assigned a unique number, the non-frequent words that appear in the question-answer list is replaced with the numeric representation of the <OUT> token.

B. Model Building

This study makes use of the seq2seq framework proposed by Ilya Sutskever, 2014. RNN model serves the base for this structure; however, since the vanilla implementation of RNN suffers from the vanishing gradient problem, the Long Short-Term Memory (LSTM) recurrent neural network variant is preferred for forming a conversational model. The Seq2Seq framework proposed by Ilya Sutskever, 2014 comprises two LSTM, one for encoding and one for decoding [20]. The seq2seq model takes as input sequence, process it one word at a time, and generates an output sequence one word at a time. Neha Atul et al., 2018, represents the mathematical description of the seq2seq model as follows: Given a sequence of input X one at a time to seq2seq encoder, it converts the input to fixed-size vector c [13]. With c as input, the decoder then predicts the probability of the output sequence Y. Maximizing the generation probability of Y conditioned on X, the objective function mentioned by the authors is shown in equation below.

$$\rho(y_1 .....,y_{T'}|x_1.....,x_T) = \prod_{t=1}^{T'} \rho(y_t|c, y_1.....,y_{t-1}) \quad (1)$$

Fig. 1 below represents the general architecture of a seq2seq model. In the figure, the encoder takes as input the text “How is the food” one word at a time and generates a context vector. The context vector is then passed to the decoder, which maps the input words to the output words. For this study, the model is build using the Tensor flow library.

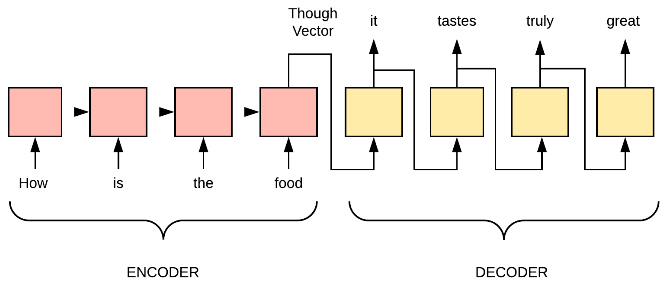

Fig. 1. Seq2Seq Model

V. Prototype

The below implementation is referenced from the work and online courses of Hadelin de Ponteves and Jordan Sauchuk.

In this study, as we are training the model on two different datasets, each section of the data preprocessing step is implemented as a python class, with section output stored in the instance variable of the class. This enables us to retrieve the output of both the dataset at any stage of the implementation.

A. Model Encoder

We define the seq2seq encoder by first defining a basic LSTM using the BasicLSTMCell library of tensor flow. A dropout of 0.5 is applied on the LSTM cell, and then the LSTM cells are composed sequentially using the MultiRNNCell function of tensor flow. The bidirectional_dynamic_rnn function is then used with the sequential LSTM cells in both forward and backward directions to generate the encoder state and encoder output.

B. Model Decoder

The training and testing decoder are defined separately. On both training and testing decoder, attention mechanism is applied so the seq2seq model can operate better on long sequences. The training and testing decoder are described using the attention_decoder_fn_train function and the attention_decoder_fn_inference function of tensor flow library. Both decoders take encoder states as input, along with attention variables. Both the established decoder are then passed to the dynamic_rnn_decoder defined in the seq2seq library of tensor flow to generate the decoder train prediction and test prediction. Also, before returning the training decoder to the user-defined seq2seq function, output dropout is applied to it, and it is fed to fully connected layers.

C. Seq2Seq Model

A seq2seq function is defined representing the seq2seq model. This function internally calls the encoder and training and testing decoder function to get the training and testing predictions.

The training data utilized to train any conversational models are generally present in the form of natural human conversation. In employing a neural network like seq2seq to the natural language task, each word in the training data has to transform into a numerical representation as a part of data preprocessing [21]. Word embedding techniques assist in converting each word into a vector of real numbers. Pre-trained models are also available for converting words to vectors. For this study, we are using the tensor flow embed_sequence layer to perform embedding on the input data before passing it to the encoder.

D. Model Training and Testing

The model is trained by splitting the inputs into batches and feeding the batches to the model for a given number of epochs. During training, the model is also fed with answers along with questions so that the model can learn through backpropagation. To ensure that the model does not overfit the training data is divided into training and validation. The model is run on the validation dataset every 100th batch. Training and validation loss are calculated using Adam optimizer. If the validation loss improves, the model weights are saved in a local file, and the training continues. If the model performs continuously inadequate on the validation set, then early stopping is applied, indicating that the model cannot perform any better. The weight file saved is then used to perform testing on the model. The model is evaluated based on human judgment. Table 1 below represents the hyperparameters used for training the model.

Currently due to the limited GPU and computation power the chatbot takes a considerable amount of time for training. As a part of this experiment we trained the chatbot on ten epochs. However the results obtained were not good enough, during testing for every question the chatbot provided a single answer.

TABLE I
HYPERPARAMETERS

Hyperparameters	**Value**
epoch	50
Batch Size	64
Learning Rate	0.01
Optimizer	Adam
Emdedding Size	512
LSTM Input Size	512
Layer Count	3
Keep Prob	0.5

VI. CONCLUSION

In this paper, an attempt has been made to understand the importance of a neural network-based chatbot system for movie-related queries. With the rise of the neural network and NLP, it is possible to extend the chatbot to automate other critical problems. The seq2seq model presented in the paper was build using the Tensor flow library. The performance of the model was confined due to limited training; however, the study showcases the usage of the data-driven approach for chatbot building. Notable performance can be achieved with powerful computational resources and making modifications to the training hyperparameters. Additionally, The textual dataset passed to the model was preprocessed using user-defined functions. Advance functions in libraries like NLTK and sklearn give us another area to explore to improve the data preprocessing.

ACKNOWLEDGMENT

I want thank Dr. Sabah Mohammed for his encouragement and supervision throughout this research project. This research work is part of the COMP5800 Research Methodology Course at Computer Science, Lakehead University, Winter2020, supervised by Prof. Dr. Sabah Mohammed.

REFERENCES

[1] Dwivedi, Sanjay K., and Vaishali Singh. "Research and reviews in question answering system." Procedia Technology 10 (2013): 417-424.
[2] Wei, Zhang, Zhang Xuan, and Chen Junjie. "Design and implementation of influenza Question Answering System based on multi-strategies." In 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE), vol. 1, pp. 720-723. IEEE, 2012.
[3] Ferrucci, David, Eric Nyberg, James Allan, Ken Barker, Eric Brown, Jennifer Chu-Carroll, Arthur Ciccolo et al. "Towards the open advancement of question answering systems." IBM, Armonk, NY, IBM Res. Rep (2009): 45.
[4] Menaha, R., A. Udhaya Surya, K. Nandhni, and M. Ishwarya. "Question answering system using web snippets." In 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), pp. 387-390. IEEE, 2017.
[5] Kahaduwa, Hasangi, Dilshan Pathirana, Pathum Liyana Arachchi, Vishma Dias, Surangika Ranathunga, and Upali Kohomban. "Question Answering system for the travel domain." In 2017 Moratuwa Engineering Research Conference (MERCon), pp. 449-454. IEEE, 2017.
[6] Ranjan, Prakash, and Rakesh Chandra Balabantaray. "Question answering system for factoid based question." In 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), pp. 221-224. IEEE, 2016.
[7] Wei, Zhang, Zhang Xuan, and Chen Junjie. "Design and implementation of influenza Question Answering System based on multi-strategies." In 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE), vol. 1, pp. 720-723. IEEE, 2012.
[8] Xiao, Linlong, Nanzhi Wang, and Guocai Yang. "A Reading Comprehension Style Question Answering Model Based On Attention Mechanism." In 2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 1-4. IEEE, 2018.
[9] Baradaran, Razieh, Razieh Ghiasi, and Hossein Amirkhani. "A Survey on Machine Reading Comprehension Systems." arXiv preprint arXiv:2001.01582 (2020).
[10] Lu, Wenpeng, Jinyong Cheng, and Qingbo Yang. "Question answering system based on web." In 2012 Fifth International Conference on Intelligent Computation Technology and Automation, pp. 573-576. Ieee, 2012.

[11] Wei, Chen, Zhichen Yu, and Simon Fong. ”How to build a chatbot: Chatbot framework and its capabilities.” In Proceedings of the 2018 10th International Conference on Machine Learning and Computing, pp. 369-373. 2018.
[12] Nuruzzaman, Mohammad, and Omar Khadeer Hussain. ”A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks.” In 2018 IEEE 15th International Conference on e-Business Engineering (ICEBE), pp. 54-61. IEEE, 2018.
[13] Godse, Neha Atul, Shaunak Deodhar, Shubhangi Raut, and Pranjali Jagdale. ”Implementation of Chatbot for ITSM Application using IBM Watson.” In 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), pp. 1-5. IEEE, 2018.
[14] Santoso, Heru Agus, Nurul Anisa Sri Winarsih, Edy Mulyanto, Septian Enggar Sukmana, Supriadi Rustad, Muhammad Syaifur Rohman, Adhitya Nugraha, and Fahri Firdausillah. ”Dinus Intelligent Assistance (DINA) Chatbot for University Admission Services.” In 2018 International Seminar on Application for Technology of Information and Communication, pp. 417-423. IEEE, 2018.
[15] Hristidis, Vagelis. ”Chatbot technologies and challenges.” In 2018 First International Conference on Artificial Intelligence for Industries (AI4I), pp. 126-126. IEEE, 2018.
[16] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. ”Sequence to sequence learning with neural networks.” In Advances in neural information processing systems, pp. 3104-3112. 2014.
[17] Vinyals, Oriol, and Quoc Le. ”A neural conversational model.” arXiv preprint arXiv:1506.05869 (2015).
[18] K.Jwala, G.N.V.G Sirisha, and G.V.Padma Raju, “Developing a Chatbot using Machine Learning” ISSN: 2277-3878, Volume-8 Issue-1S3, IJRTE, 2019.
[19] Muangkammuen, Panitan, Narong Intiruk, and Kanda Runapongsa Saikaew. ”Automated thai-faq chatbot using rnn-lstm.” In 2018 22nd International Computer Science and Engineering Conference (ICSEC), pp. 1-4. IEEE, 2018.
[20] Hayat, Aadil, and Masare Akshay Sunil. ”A Neural Conversational Model.”
[21] Csaky, Richard. ”Deep learning based chatbot models.” arXiv preprint arXiv:1908.08835 (2019).
[22] Nguyen, Huyen, David Morales, and Tessera Chin. ”A neural chatbot with personality.” (2017).
[23] Deep Learning and NLP A-Z™: How to create a ChatBot https://www.udemy.com/course/chatbot/learn/lecture/

APPENDIX

A. Data Reshaping Cornell Dataset

```
class DataReshapingCornell:

  #Name: __init__
  #Description: Initializes the dataset and quesion
    answer list
  #Input: None
  #Ouput: None
  def __init__(self,datasetLines,datasetConversation
    ):
    self.datasetLines=datasetLines
    self.datasetConversation=datasetConversation
    self.question=[]
    self.answer=[]

  #Name: getConversationList
  #Description: Get list of conversations from
    movie_conversations.txt (for eg ['L194', 'L195',
     'L196', 'L197'])
  #Input: None
  #Ouput: List Consisting list of conversations
  def getConversationList(self):
    conversation_list=[]
    for lines in self.datasetConversation:
      listString=lines.split("+++$+++")[-1].strip()
      listString=listString[1:-1]
      listString=listString.replace("'","")
      listString=listString.replace(" ","")
      conversation_list.append(listString.split(",")
    )
    return  conversation_list

  #Name: getConversationIdDict
  #Description: Map Conversation ID to the
    Conversation from movie_lines.txt
  #Input: None
  #Ouput: Dictionary {Conversation id : Coversation}
  def getConversationIdDict(self):
    conversation_id_mapping={}
    for lines in self.datasetLines:
      conversation=lines.split("+++$+++")
      conversation_id_mapping[conversation[0].strip
    ()]=conversation[-1].strip()
    return conversation_id_mapping

  #Name: questionAnswerFormation
  #Description: Create a list of question and answer
     with one to one mapping
  #Input: None
  #Ouput: Question and Answer List formed
  def questionAnswerFormation(self,conversation_list
    ,conversation_id_mapping):
    conversation_list_len=len(conversation_list)
    for itemslist in range(0, conversation_list_len)
    :
      sub_conversation_list=conversation_list[
    itemslist]
      for item in range(0,len(sub_conversation_list)
    -1,):
        questionId=sub_conversation_list[item]
        answerId=sub_conversation_list[item+1]
        self.question.append(conversation_id_mapping
    [questionId])
        self.answer.append(conversation_id_mapping[
    answerId])

  #Name: beginDataFormationCornell
  #Description: Starting point for reshaping cornell
     movie dataset
  #Input: None
  #Ouput: None
  def beginDataFormationCornell(self):
    conversation_list=self.getConversationList()
    conversation_id_mapping=self.
    getConversationIdDict()
    self.questionAnswerFormation(conversation_list,
    conversation_id_mapping)
```

Listing 1. Data Reshaping Cornell Dataset

B. Seq2Seq Model

```
def seq2seq_model(inputs, targets, keep_prob,
    batch_size, sequence_length, len_vocabwordid,
    encoder_embedding_size, decoder_embedding_size,
    rnn_size, layers_count, vocabwordid):
    # embedding the encoder input
    embedded_encoder_input = tf.contrib.layers.
    embed_sequence(inputs,len_vocabwordid + 1,
    encoder_embedding_size,initializer=tf.
    random_uniform_initializer(0, 1))

    #getting the output of the encoder
    encoder_state = encoder(embedded_encoder_input,
    rnn_size, layers_count, keep_prob,
    sequence_length)

    #Modify the answers with <sos>
    preprocessed_targets = target_preprocessing(
    targets, vocabwordid, batch_size)

    #define a decoder matrix for the embedding
```

```
decoder_embeddings_matrix = tf.Variable(tf.
random_uniform([len_vocabwordid + 1,
decoder_embedding_size], 0, 1))

#perform lookup on tensors list
embedded_decoder_input = tf.nn.embedding_lookup(
decoder_embeddings_matrix, preprocessed_targets)

#get the predictions
training_inference, test_inference = decoder(
embedded_decoder_input,decoder_embeddings_matrix
,encoder_state,len_vocabwordid,sequence_length,
rnn_size,layers_count,vocabwordid,keep_prob,
batch_size)

return training_inference, test_inference
```

Listing 2. Seq2Seq Model