





MARGEN: Marathi Question Answering Generative Conversation Model

Satish V. Bhalshankar^(✉)  and Ratnadeep R. Deshmukh 

Department of Computer Science and IT, Dr. Babasaheb Ambedkar Marathwada University,
Aurangabad, MH, India

sbhalshankar.mgmtsci@bamu.ac.in

Abstract. The conversational system aka chatbot market capture was worth USD 526 million in 2021 around the world. The innovations created such as Machine learning, Deep learning, Natural Language Processing (NLP), and Big data analytics have given a modern speed-quicken fuel to Artificial Intelligence. Well, a generative chatbot could be an exceptionally effective and shrewd conversational system as distant as its learning component is concerned. They can be trained from scratch like a newborn child by using Deep Learning techniques. GPT-1,2,3 is well known for English Language NLP tasks whereas IndicBART is also moving ahead for 11 Indian Languages for the NLU tasks. With the DL techniques like RNN, LSTM, and SGD, we conducted an in-depth survey of recent deep learning conversational models available on open-source portals, examining over 25 deep learning models for conversational systems before proposing our model. "MARGEN" is a proposed model which means the user can get the system's reply with proper generative answers for the query in Marathi text and/or audio speech formats.

Keywords: Generative Chatbot · GPT · IndicBART · AI · NLP · RNN · LSTM · Dataset

1 Introduction

Human Intelligence, which is nature's gift to do human's necessary actions and the interest of shrewdly computational machines, has become a critical attempt of humankind. Machines have been a few minds-blowing innovative propels within this century's field of Artificial Insights [1]. These progressive revelations are a result of decades of investigation on the basis of trials. Nowadays, computational linguistics and computer science are changing numerous perspectives of businesses and human life[4]. With NLP innovation, machines now learn the contents and make sense of Human Dialect. It has opened an interesting and important chapter in Machine learning [3].

The worldwide chatbot market measure was worth USD 526 million in 2021. It is anticipated to reach USD 3,7023 million by 2030, growing at a CAGR of 24.1% amid the estimated period (2022–2030) [2].

A chatbot system uses conversational artificial intelligence (AI) in innovatively to re-enact a discussion (or chat) with a client by means of informing apps, websites,

mobile apps, or the phone [1]. In real-time, client interactions are prepared through rule-based dialect applications that perform live chat capacities. One of the foremost noticeable components of chatbots is that, in contrast to applications, they don't have to be downloaded, and don't utilize up any memory on the phone. Another advantage is that numerous bots can be combined into one discussion simultaneously.

A chatbot can progress and engage in client's intelligent discussion whereas requiring less intervention from human representatives [5]. It evacuates the boundaries to giving benefit to clients that can show up when there's a more prominent request than the capacity to meet it. Clients can get moment responses to their questions instead of hold up on hold. By bringing down the bother that client's involvement with a company's administrations, brand encounters can be moved forward. Chatbot virtual associates are getting to be progressively prevalent among clients in business-to-consumer (B2C) and business-to-business (B2B) settings.[2, 5] These colleagues are utilized to total basic assignments. The usage of chatbot collaborators comes about in a diminish in overhead costs, more effectively utilize of the time of support staff, and the capacity for businesses to supply client benefit exterior of normal trade hours.[6].

Based on end-users, the worldwide chatbot market has been classified into Huge, Medium, and Little Undertakings. Huge Undertakings are anticipated to produce the most noteworthy market share, developing a CAGR of 24.1% amid the figure period. Based on the commerce demonstrate, the worldwide chatbot market has been classified into Bot for Benefit, Bot for Social Media, Bot for Payments/Order Handling, Bot for Promoting, and Others [7].

For most of us, the extreme extravagance would be a collaborator who continuously listens in for your call, expects your each require, and takes activity when essential. Personal advanced assistants like Siri from Apple, Alexa from Amazon, Microsoft's Cortana or Collaborator from Google are at the cutting edge of innovation of voice acknowledgment and artificial intelligence [2, 8].

It could be a well-known reality that innovation is advancing exceptionally quick. As a result, the run of innovation increments day by day and comes about in low-cost computing. The innovations creating such as Machine learning, Deep learning, Natural Language processing (NLP), and Big data analytics have given a modern speed quickening fuel to Artificial Intelligence [9, 10]. As a result of which it is conceivable to execute conversational Interface Intellectuals. These Shrewdly (stacked with logical ML/DL models) conversational interfacing, which utilize Machine Learning, Profound Learning as their spine. It is pointless for these Chatbots Applications continuously to be textual. These intelligent can be Voice and Visual based [11].

1.1 Generative Conversational System: A Brief Idea

A generation-based conversational system produces reply reactions rather than selecting them from the fundamental ANN model [12]. A sequence-to-sequence to sequence (S2S) system was created to produce answers or reply to users' input request [16]. In fact, a sequence-to-sequence system which encodes a message with artificial neural network and creates reply utilizing another repetitive neural network with support of attention mechanism [26].

Well, a generative chatbot could be an exceptionally effective and shrewd conversational system as distant as its learning component is concerned [14]. They can be trained from scratch like a newborn child by using Deep Learning techniques.

With these machine learning techniques, a conversational system can learn through the information it experiences as well as the discussions it has with individuals and indeed other pretrained models. Fundamentally, there are two types of conversational systems, one is the closed domain, meaning you're constrained to a predefined (or closed) set of questions, whereas another is an open domain, or a Generative conversational system [17].

The vital distinction between these two is that the closed domain conversational system can as it were reacted to a list of predefined questions with trained answers, whereas the generative domain conversational system can produce reactions on its claim by making utilize of the past information the chatbot put away as well as the past intelligent consequently the title generative chatbot [15].

In the event that they don't know something, they will ordinarily state that and inquire for more data. This is often how they learn. The chatbots are made so they can make an awareness of them possess utilizing Profound Learning forms; it can be instructed how to conversation to individuals through content [17]. A Profound Learning chatbot can moreover learn from movie dialogues, play scripts, expositions, and literature. However, perfect way" > the most ideal way for a generative chatbot to memorize is human intelligent; the more an AI chatbot talks with a human, the more it can learn around the world, different subjects, and the craftsmanship of having discussions [27, 28].

1.1.1 Preparation of Generative Chatbot Conversational System

To Construct a Generative Conversational System Gather and Get ready Data.

The exceptionally to begin with step in building the generative Conversational System is to begin with gather preexisting information readily available on web. Developer wishes to have tons of information accessible in their capacity from human intuitive or other elective ways, such as play scripts or movie dialogues [16, 29]. However, the foremost ideal strategy will be to gather tons of data from human intuitive. Pay attention to the points of interest when collecting the information and make beyond any doubt that the information collected would be the foremost ideal for conversational system to memorize from neural network.

The Data Reshaping for Compilation Step. This is optional and depends if the data is varied during collection. But on the off chance that the data isn't changed or isolated, at that point developer might got to compile it sometime recently it gets considered to utilize for generative conversational system. Developer needs to add labels to the set of data found during the interactions, make sets, and label who the speaker is and who is reacting to the speaker within the discussions. This will offer assistance the system to come up with a response.

Pre-Process the Conversational System. Pre-processing the chatbot implies creating solid yields designers have to be include semantic acknowledgment in conversational framework so that it can work effectively, I.e., including linguistic use into framework

awareness so it can maintain a strategic distance from the spelling blunders and get it and translate the human blunders it experiences all through its conversation. After including all the requisite tools into the chatbot's neural organize so it can study the information more effectively [10]. The common steps related with pre-processing are tokenizing, stemming, and lemmatizing. Developers have to be utilized online assets for all the over steps.

Creating Word Vector. To construct a generative conversational system's word vectors is important step [10]. Word vectors are fundamental to utilize when the chatbot framework experiences words that as often as possible appear on Web or online media however are not a portion of the official dialect databases. For e.g. LOL, WTH, etc. The developer can either utilize pre-trained vectors to achieve the errand or can make them possess word vectors. The moment strategy is more time-consuming however as a better pay-off as that way, chatbot will be able to handle the discussions more efficiently. A Word2Vec demonstrate for this errand is more reasonable. It can produce word vectors by recognizing the setting in which the words are utilized. Hence by learning the utilize of words in sentences, the chatbot produces its vectors [20].

Creating a Conversational System's Model. The following step within the creation of generative conversational system is to begin creating chatbot demonstration. Software engineers developed a Seq2Seq demonstration to achieve the errand. Developer can moreover utilize TensorFlow, which is an open-source programming library and is accessible without charge to everyone [49].

Testing the Conversational System. Developers got to perform tests on the conversational system once get ready it, to check whether it produces the specified comes about [13]. Track the method, i.e., check how the conversational demonstrate carries on and check for extra blunders within the chatbot's behavior. The developer should check what and which tokens the chatbot returns and the input the framework sends back at diverse focuses within the preparing prepare [45]. Prepare the conversational framework drearily, so it can in the long run send back complex strings. Initially, the information it'll return will comprise of basic strings, but the chatbot will produce more complex sentences as the method takes its stream and the chatbot learns.

Adding the Model to Applications. After testing the seq2seq generative model, the following step is to coordinated it with an application. When finding the proper application to integrate the demonstrate with, designer ought to choose a stage with sufficient activity so it can connect with individuals with learning from them. One of the fastest ways to attain this objective is by choosing Facebook or courier for the assignment. It also offers a basic strategy of including a chatbot and is an ideal stage for the conversational framework to memorize human interactions [40, 42].

1.2 GPT Models for English Language and IndicBART for Marathi

1.2.1 Gpt-1

OpenAI had launched first generative model based on NLP concept, i.e. GPT -1. GPT -1 was trained on a colossal Books-Corpus dataset, GPT-1 generative dialect model was

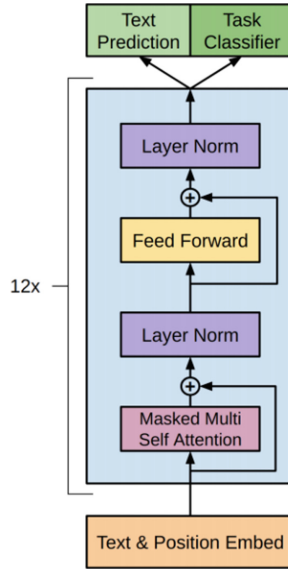


Fig. 1. GPT -1 Model

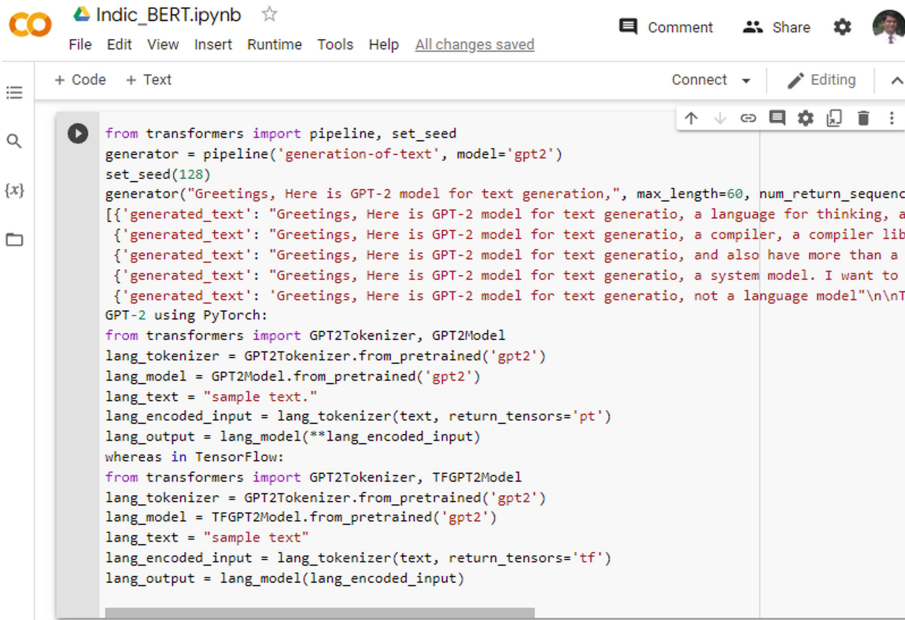
prepared for memorize huge extended dependences and obtained the requisite information on a differing corpus of coterminal content & large extends. As shown in the Fig. 1, GPT-1 applied the transformer decoder design of 12 layers with a self-attention component for training. With Exchange learning as its base GPT got to be an effective facilitator to perform characteristic dialect handling errands with exceptionally small fine-tuning. It produced trails for other language models which seem to assist upgrade its potential in training with huge datasets along with defined parameters.[18, 19].

1.2.2 GPT-2

GPT-2 could be a transformer based generative model pretrained on a corpus of contents of the English language in a supervised design which implies it was pretrained on the basic writings as it were, without humans labelling. Actually, input sequence are arrangements of uninterrupted content of a particular length and the targets are the same sequence, moved to the proper token. The GPT-2 used inside a mask-mechanism to create beyond any doubt the expectations for the token i as it were employments the inputs from 1 to i but not long-term tokens. This way, the GPT-2 model shown in Fig. 3 learnt an internal representation of the English dialect that can at that point be utilized to extricate highlights valuable for downstream errands. The demonstrate is best at what it was pretrained for be that as it may, which is producing writings from a provoke [20, 21, 22].

The proper calling the gpt2 model is as following Fig. 2.

There is an unofficial training dataset utilized for this GPT-2 model. It contains a part of web-based content, which is mostly biased. Since GPT-2 don't recognize reality from fiction, use-cases that require the created content to be genuine are meaningless. The



```

from transformers import pipeline, set_seed
generator = pipeline('generation-of-text', model='gpt2')
set_seed(128)
generator("Greetings, Here is GPT-2 model for text generation,", max_length=60, num_return_sequences=1)
generator("Greetings, Here is GPT-2 model for text generation, a language for thinking, a compiler, a compiler lib")
generator("Greetings, Here is GPT-2 model for text generation, and also have more than a")
generator("Greetings, Here is GPT-2 model for text generation, a system model. I want to")
generator("Greetings, Here is GPT-2 model for text generation, not a language model")
GPT-2 using PyTorch:
from transformers import GPT2Tokenizer, GPT2Model
lang_tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
lang_model = GPT2Model.from_pretrained('gpt2')
lang_text = "sample text."
lang_encoded_input = lang_tokenizer(text, return_tensors='pt')
lang_output = lang_model(*lang_encoded_input)
whereas in TensorFlow:
from transformers import GPT2Tokenizer, TFPGPT2Model
lang_tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
lang_model = TFPGPT2Model.from_pretrained('gpt2')
lang_text = "sample text"
lang_encoded_input = lang_tokenizer(text, return_tensors='tf')
lang_output = lang_model(lang_encoded_input)

```

Fig. 2. Procedure to use GPT -2 Model

OpenAI group needed to prepare GPT-2 on a corpus as big as believable. To construct such dataset, developers tried to scrap all the web pages from Reddit and Wikipedia. The coming about dataset (called WebText) weights 40GB of writings but has not been freely discharged. Engineer can discover a list of the beat 1,000 spaces display on Hugging Face [22].

GPT-2 preprocessed utilizing the writings are tokenized employing a byte-level form of Byte Match Encoding (BPE) (for Unicode characters) and a lexicon estimate of 50,257. The inputs are arrangements of 1024 sequential tokens. The bigger show was prepared on 256 cloud TPU v3 centers [23].

1.2.3 GPT-3

GPT-3 is the recent form of the Generative pre-training Model series. It could be an enormous dialect forecast and generation model created by OpenAI capable of creating long arrangements of the initial as well as old content. GPT-3 became the OpenAI's breakthrough AI dialect program. It may be an application that can naturally create passages so one of a kind that it nearly sounds as in the event that an individual composed them. GPT-3 program is right now accessible with confined get to through an API on the cloud, and get to is required to investigate the instrument. It has made a few captivating applications since its dispatch. Its critical advantage is its measure, it contains around 175 billion parameters and is 100 times bigger than GPT-2. It is prepared upon a 500-billion-word information set i.e., Common Crawl collected from the web and available repositories [24].

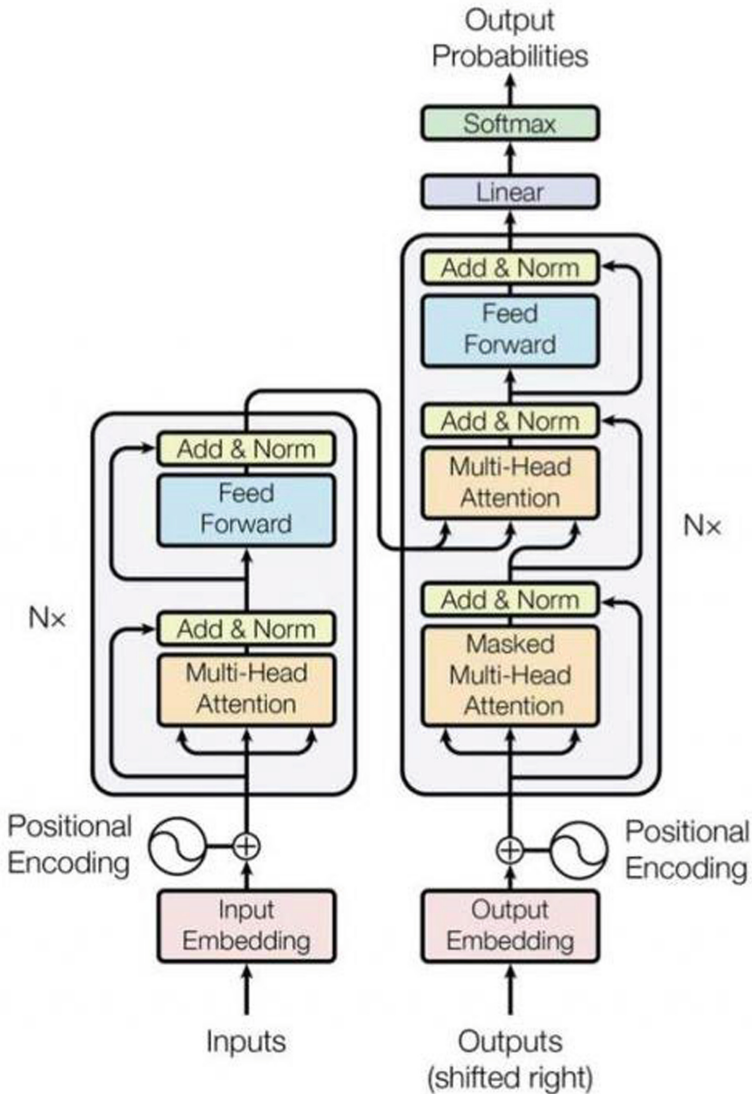


Fig. 3. GPT -2 Transformer Model

Its other critical and shocking capacity is to perform basic number juggling issues, counting composing code pieces and execute cleverly errands. The comes about are quicker reaction time and exactness permitting NLP models to advantage trade by viably and reliably keeping up best hones and decreasing human mistakes. The reason of GPT-3 was to form dialect handling more effective and speedier than its past forms and without any uncommon tuning. Most of the past dialect handling models (such as BERT) require in-depth fine-tuning with thousands of illustrations to educate the show how to

Table 1. Comparison between GPT 1, 2, and 3

	GPT 1	GPT 2	GPT 3
Parameters Used	117 M	1.5 B	175 B
No. of Layers of Decoder	12	48	96
Size of Context Token	512	1024	2048
No. of Hidden Layer	768	1600	12288
Total Batch Size	64	512	3.2M

Table 2. Indian Language Supported by IndicBART and their processed tokens

Language	No. of Tokens
Assamese	36.9 million
Bengali	815 million
English	1.34 billion
Gujrati	724 million
Hindi	1.84 billion
Kannada	712 million
Malayalam	767 million
Marathi	560 million
Odissi	104 million
Panjabi	814 million
Tamil	549 million
Telugu	671 million
All	8.9 billion

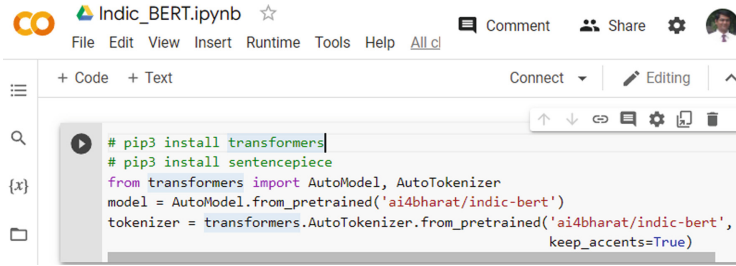
perform downstream errands. With GPT-3 clients can dispose of the fine-tuning step. The contrast between the three GPT models is their measure. The unique Transformer Demonstrate had around 110 million parameters. GPT-1 received the estimate and with GPT-2 the number of parameters was improved to 1.5 billion. With GPT-3, the number of parameters was boosted to 175 billion, making it the biggest artificial neural network [24, 25]. (Shown in Table 1) (Table 2 and 3).

1.2.4 IndicBART: Indian Model for Text Generation

IndicBART, a multilingual pre-trained sequence to sequence model particularly prepared for Indic dialects, which are talked by more than a billion users. It bolsters English and 11 Indian dialects counting 7 Indo-Aryan i.e. Assamese, Bengali, Gujarati, Hindi,

Table 3. Selection Criteria

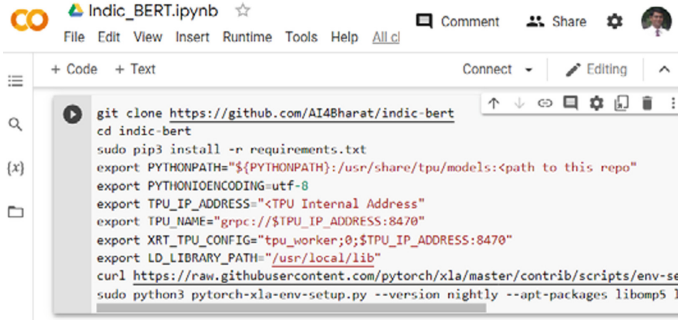
Inclusion Criteria (IC)	Exclusion Criteria (EC)
IC1: Source Code developed in python	EC1: Excluded other than python due to limitations of pre-defined libraries
IC2: Deep Learning Techniques	EC2: Implementation other than Python Language
IC3: Datasets Available	EC3: Not having dataset
IC4: Published between 2018 and 2021	



```

# pip3 install transformers
# pip3 install sentencepiece
from transformers import AutoModel, AutoTokenizer
model = AutoModel.from_pretrained('ai4bharat/indic-bert')
tokenizer = transformers.AutoTokenizer.from_pretrained('ai4bharat/indic-bert',
                                                    keep_accents=True)

```

Fig. 4. Installation and Creating model using IndicBART


```

git clone https://github.com/AI4Bharat/indic-bert
cd indic-bert
sudo pip3 install -r requirements.txt
export PYTHONPATH="${PYTHONPATH}:/usr/share/tpu/models:<path to this repo>"
export PYTHONIOENCODING=utf-8
export TPU_IP_ADDRESS="<TPU Internal Address>"
export TPU_NAME="grpc://$TPU_IP_ADDRESS:8470"
export XRT_TPU_CONFIG="tpu_worker;0;$TPU_IP_ADDRESS:8470"
export LD_LIBRARY_PATH="/usr/local/lib"
curl https://raw.githubusercontent.com/pytorch/xla/master/contrib/scripts/env-se
sudo python3 pytorch-xla-env-setup.py --version nightly --apt-packages libomp5 l

```

Fig. 5. Exporting the necessary libraries supported by IndicBART

Marathi, Oriya, Punjabi and 4 Dravidian i.e. Kannada, Malayalam, Tamil, Telugu languages [30, 31, 32]. It is pre-trained on AI4Bharat's monolingual corpus. The corpus has the taking after conveyance of languages:

The easiest way to use Indic BERT is through the Hugging face transformers library. It can be simply loaded like this (shown in Fig. 4 and 5):

The following viewpoints of the IndicBART model contribute to its solid execution and expanded dialect scope within the Indic Language models [33, 34], whereas being profoundly compact:

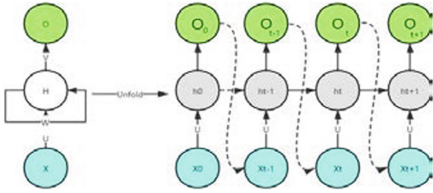


Fig. 6. Structure of Recurrent Neural Network

1. It is prepared on a little set of related dialects, which diminishes model capacity prerequisites. Besides, accessible model capacity is viably utilized, since transfer learning works when dialects share etymological highlights and information speaks to shared themes.
2. It is prepared on publicly accessible Indic dialect corpora, IndicCorp, which incorporates huge, high-quality news crawls for Indian dialects as well as English substance from Indian websites- hence being an agent of Indian English and themes.
3. Currently, We are trying to map the question’s text with an expected summarized reply using Indic dialect information.

That’s why in the following proposed model, as of now we are investigating with the help of IndicBART, a multilingual, sequence-to-sequence pre-trained model centering on 11 Indic dialects for Marathi question answers summarization. Diverse from existing pre-trained models, IndicBART utilizes the orthographic likeness between Indic scripts to progress exchange learning between comparable Indic languages [37]. We wish to assess IndicBART on these NLG tasks: Neural Machine Interpretation (NMT) [32, 35] and summarization for a reply. Our tests on NMT for the Marathi language question-answer pairs and summarization of the same in the MARathi dialect utilizing fine-tuning approach [36, 38, 39].

2 Deep Learning Techniques for Conversational System

Most deep learning-based chatbot innovations are built on the Encoder-Decoder deep learning model system.

2.1 Recurrent Neural Network Structure

For most neural networks, expectation of the input data to be uncorrelated, but in the chatbot systems, developers need to consider the relationship between the data and neural networks [43]. The performance of a RNN always depends on both the current input and the state of the past moment’s information. Representation of the time input component allows the model to make efficient use of contextual information [44] The structure of the RNN network is shown in following Fig. 6.

The activation formula is as follows:

$$h_t = f(Wx_t + Uh_{t-1} + b) \tag{1}$$

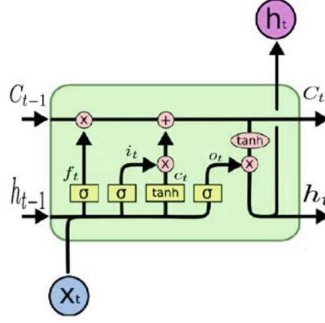


Fig. 7. LSTM basic structure

where W , U and B are the weight parameters of the show, h_t state to the hidden layer at time t , x_t means to the input at time stamp t , as well as f is a nonlinear enactment formula. In practical implementation, there were a few extended input groupings of chatbots. As the ultimate derivative will incorporate the product of gradients in every step among back propagation determination [43], which leads to the issue of gradient vanishing in this RNN network structure [43].

2.2 Long Short-Term Memory Structure

The algorithm structure of RNN is successful to manage with time series problems, but there's an issue of gradient vanishing. Hence, the development of LSTM (Long Short-Term Memory) can illuminate the issue that RNN inputs long content groupings, and the information at the back conclusion of the arrangement will cover up the data at the front end [47], coming about within the failure to completely express the in general data of long content sequences.

Long Short Term Memory Structure shown in Fig. 7 is made up of an arrangement of timing modules [47]. A LSTM neuron incorporates three entryways or gateways: first is forgetting gate, second is input gate and last is output gate. The gates are utilized to prepare and channelize data, taking off the specified information behind and disposing of the pointless data [48]. LSTM is communicated as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (3)$$

$$C_t = f_t * C_{t-1} + i_t * \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (4)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * \tanh(c_t) \quad (6)$$

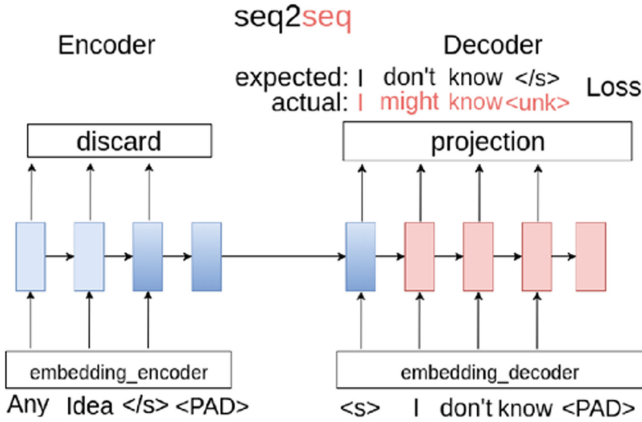


Fig. 8. The Seq-2-Seq Model for Chatbot System

where $ht-1$ is the preceding neuron's output and ht stands for hidden state. xt stand for the current cell 's input; Wf , Wi , WC , $W0$ are weight matrices of LSTM; bf , bi , bC , $b0$ are the biases of the LSTM;

f_t which represents forgetting gate's output and i_t represents input gate's output.
 $\sigma(\cdot)$ represents the sigmoid activation function.

2.3 Sequence to Sequence Structure

In the RNN structure, the lengths of the input and output sequences must be balanced, but in practical implementations the size of the input as well as output groups are different. Therefore, the Seq2Seq model is proposed to solve the issue which was used for machine translation tasks along with outperformed proven traditional machine translation. This is where the Seq2Seq pattern in the chatbot comes in [45, 48]. The encoder module is supposed to encode the data input sequence into a fixed length proper labelling vector C , whereas the decoder module reliably interprets this vector C as the target or output sequence [48].

For Seq-2-Seq, Encoder and Decoder can receive a variety of diverse neural network structures shown in Fig. 8. In the dataset, the data should be characterized by $\langle H, Y \rangle$, where H represents the user's query and Y represents the expected response associated with the query, as follows:

$$H = \langle h_1, h_2, h_3, \dots, h_t \rangle$$

$$Y = \langle y_1, y_2, y_3, \dots, y_k \rangle$$

Where t and k represent the maximum length of question H and reply Y respectively.

$$p(y_1 \dots y_{N'} | x_1, x_2, x_n) = \prod_{i=1}^{N'} p(y_i | v, y_1, y_2, y_{i-1}) \quad (7)$$

The main target of the above formula is to predict the conditional probability of $P(y_1, y_2, y_3, \dots, y_n | x_1, x_2, x_3, \dots, x_n)$, where $x_1, x_2, x_3, \dots, x_n$ is the input sequence and y_1, \dots, y_n 's the corresponding output sequence [46].

- **Encoder** The encoder has to convert the input sequence H which is unlimited length into the context vector C of fixed length with help of a nonlinear transformation series
- **Decoder** at the time of decoding, the output sequence y_2, \dots, y_{t-1} , and the context vector C are used to predict the next output word y_i
- **Output results** with respect to the proper and optimum approximate estimation, the conditional probability of the output sequence in a given input sequence H has to maximize, and output sequence's loss i.e., Y obtained appropriately.

The Seq2Seq model has few drawbacks. First one, the encoder module to convert the data input sequence into a fixed size vector C is mostly type of lossy compression, which makes the loss of semantic and intentional data due to compressed formation. Secondly, the data word at the vertebral side of the input sequence is supported by the data at the obverse side, and the data transfer is unidirectional, making the data more "resolvable" within the sequence.

2.4 Attention Mechanism

The attention mechanism [49] mimics the attentional characteristics human brains. The attention mechanism is more associated with sequential learning tasks, language understanding related to NLP. For the Seq2Seq model, which is Encoder and Decoder modules, and both modules can process the data with Attention Mechanism for better result. By including the Attention mechanism in the Encoder module, weights are assigned to the input data.

Encoder and Decoder within the conventional Seq2Seq model transfer data via a halfway vector, and the length of encoding vector is almost fixed, which is supposed to create of long- separate neural network of the model, that's why, for long sequence content, when input sequence given for processing, the data within the vertebral side of the sequence has to protect the data within the obverse side of the sequence [48].

Attention by holding portion of Encoder's output comes about to the input grouping, such as c_2 in Fig. 7, the preparing show specifically learns this output comes about and will in the long run be related with the corresponding output sequence [43]. A theoretical diagram of the mechanism Seq2Seq appears shown in Fig. 9.

Here s_i is state of the RNN's hidden layer at time stamp i , and the mathematical formula to calculate s_i is:

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \quad (8)$$

For every epoch output h_i , different weights are introduced here

$$c_i = \prod (j = i)^{T_x} * \llbracket a_{ij} * h_i \rrbracket \quad (9)$$

The calculation formula of a_{ij} & h_i in each hidden layer is as follows:

$$\exp(a_{ij}) / (\sum (k = 1)^{T_x} * \exp(e_{ik})) \quad (10)$$

$$e_{ij} = a(s_{i-1}, h_j) \quad (11)$$

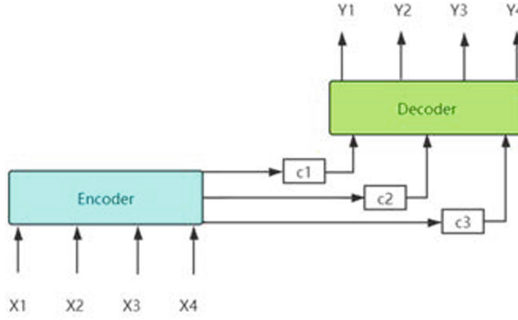


Fig. 9. Seq-2-Seq model with Attention

The function a has several implementation directions, and the following is one of them.

$$e_{ij} = v^T \tanh (W_s s_{i-1} + W_h h_i) \quad (12)$$

Among them, v , W_s , W_h , i.e. different weights, offset terms, and embedding layer parameters for the encoder and the decoder are all model parameters that must be learned simultaneously.[45].

3 Reviewed Generative Models Based DL Technique

After studying the DL techniques and Before proposing our model, we conducted an in-depth survey of recent deep learning models on opensource portals, examining over 25 deep learning models for conversational systems published in the period 2018–2021 years on GitHub web portal.

We inspected more than 25 distinctive deep learning conversation models which accessible on GitHub portal. Here, out of 25 we discussed 5 models in point by point. The genuine execution and cross confirmation of generative chatbot or conversational models dialogue based on different types of deep learning technique discussed in earlier. [41, 45, 47] The attention mechanism is based on the thought of shared information and investigations the quality of the show based on the effect of the ultimate reaction impact. We also checked about the predefined libraries, sample dataset, model selection, model training, accuracy and loss, and result of the each demonstrate from 6 chosen models.

The following criteria we applied for deciding the implementation trial of selected model.

We used python 3 environment, TensorFlow 2, Jupiter Notebook and cuda 11 gpu to run following models for evaluating their performance.

3.1 Generative Chatbot Using Seq2seq by Rami-Ramudu https://github.com/rami-ramudu/Generative-Chatbot-LSTM/tree/main/Python_chatbot-master

The model based on seq2seq and used a small dataset for training the model which is available on https://github.com/shubham0204/Dataset_Archives. The model structure developed by him is showing in Fig. 10:

```

In [5]: encoder_inputs = tf.keras.layers.Input(shape=( None , ))
encoder_embedding = tf.keras.layers.Embedding( VOCAB_SIZE, 200 , mask_zero=True ) (encoder_inputs)
encoder_outputs , state_h , state_c = tf.keras.layers.LSTM( 200 , return_state=True )( encoder_embedding
encoder_states = [ state_h , state_c ]

decoder_inputs = tf.keras.layers.Input(shape=( None , ))
decoder_embedding = tf.keras.layers.Embedding( VOCAB_SIZE, 200 , mask_zero=True ) (decoder_inputs)
decoder_lstm = tf.keras.layers.LSTM( 200 , return_state=True , return_sequences=True )
decoder_outputs , _ , _ = decoder_lstm ( decoder_embedding , initial_state=encoder_states )
decoder_dense = tf.keras.layers.Dense( VOCAB_SIZE , activation=tf.keras.activations.softmax )
output = decoder_dense ( decoder_outputs )

model = tf.keras.models.Model([encoder_inputs, decoder_inputs], output )
model.compile(optimizer=tf.keras.optimizers.Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

```

Fig. 10. Model Structure of Generative Chatbot designed by Rami-Ramudu

In above, Rami developed a model using sequential technique where tf algorithm used for creating neural network. In this neural network, dense network is having 200 layers & softmax function for final decision. In model, categorical_crossentropy method used for checking the model's loss and accuracy which shown in Fig. 13 whereas optimizer was sgd for expected results about the selection of reply.

He followed the steps for model development:

- Pre-processing the data
- Creating the Encoder-Decoder Model
- Training the model
- Creating model to predict answers
- Creating GUI for talking with our chatbot
- Deployment or integration with any application or website

Readers can understand after observing the model summary in Fig. 11 in detail (Fig. 12).

If observed the Fig. 10 we got idea about the model's accuracy which is approximately 65% and loss 32% showing better performance after 100 epochs.

3.2 Generative Chatbot Using Seq2seq by Aakriti CSE Undergrad, IIT Bombay https://github.com/Aakriti28/Generative_chatbot

The model based on seq2seq and trained on a small dataset composed of 8000 pairs of context and respective response. For the dataset, the data were collected from dialogues of English Movies designed by Aakriti. The algorithm and model structure developed by her is as follows (Fig. 14 and Fig. 15):

In above, Aakriti has developed a model using sequential technique where she has used SGD [33] algorithm for creating neural network. In this neural network, dense network is having 128 layers each and relu function for activation from hidden layer and softmax function for final decision. She used categorical_crossentropy method for checking the model's loss and accuracy which shown in Fig. 9 whereas optimizer was sgd for expected results about the selection of reply (Fig. 16).

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, None)]	0	[]
input_2 (InputLayer)	[(None, None)]	0	[]
embedding (Embedding)	(None, None, 200)	378800	['input_1[0][0]']
embedding_1 (Embedding)	(None, None, 200)	378800	['input_2[0][0]']
lstm (LSTM)	[(None, 200), (None, 200), (None, 200)]	320800	['embedding[0][0]']
lstm_1 (LSTM)	[(None, None, 200), (None, 200), (None, 200)]	320800	['embedding_1[0][0]', 'lstm[0][1]', 'lstm[0][2]']
dense (Dense)	(None, None, 1894)	380694	['lstm_1[0][0]']

=====

Total params: 1,779,894
Trainable params: 1,779,894
Non-trainable params: 0

Fig. 11. Model Summary of Generative Chatbot designed by Rami-Ramudu

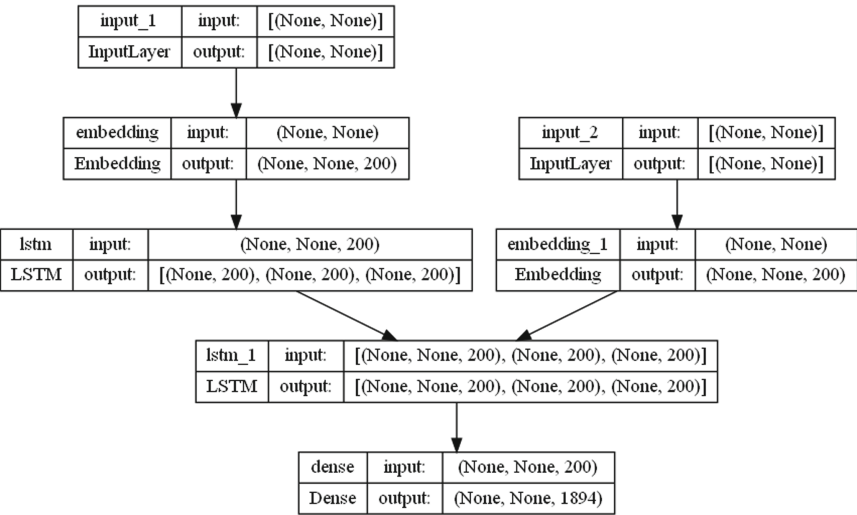


Fig. 12. Rami’s Generative Chatbot Model Summary

She followed the steps for model development:

- **Model Training:** Next, she vectorized the text data corpus by using the “Tokenizer” class and it allows us to limit the vocabulary size up to defined number. In further trained the model using “fit” method with training data and labels. We will get more detailed idea after observing the model summary in Fig. 17 and 18.

If we observed the Fig. 19 we got idea about the model’s accuracy which is approximately 98% and loss 2% showing better performance after 200 epochs.

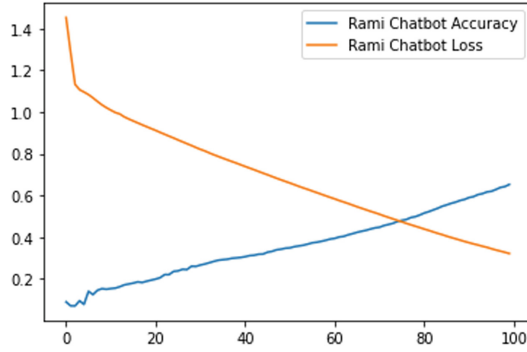


Fig. 13. Rami's Generative Chatbot Model Performance Metrics

Algorithm

```

1: Input:  $x$ : the input sequence (context text).
2: Output:  $y$ ,  $p$ : the sampled output sequence and its conditional probability  $p(y|x)$ .
3:  $p \leftarrow 1$ 
4:  $y \leftarrow []$ 
5:  $y \leftarrow \text{'BOS'}$  (symbol representing the beginning of the sentence);
6: while  $y \neq \text{'EOS'}$  (symbol representing the end of sentence) do
7:    $y \leftarrow [y, y]$ ;
8:   input  $x$  and  $y$  into the two input layers of the model
9:    $y \leftarrow$  token corresponding to the largest output of the model
10:   $p(y|x, y) \leftarrow$  the value of the larger output of the model
11:   $p \leftarrow p \times p(y|x, y)$ ;
12: end while

```

Fig. 14. Algorithm proposed by Aakriti

3.3 Python-Chatbot-Project-Using-NLTK-And-Keras by Shruti Mukherjee

Weblink: <https://github.com/shruti821/Python-Chatbot-Project-using-NLTK-and-Keras>

The model based on seq2seq and intent pairs dataset designed by Shruti Mukherjee. The model structure developed by her is as follows (Fig. 20):

In above, Shruti has developed a model using sequential technique where she also has used SGD algorithm for creating neural network. In this neural network, dense network is having 128 layers each and relu function for activation from hidden layer and softmax function for final decision. She also used categorical_crossentropy method for checking the model's loss and accuracy which shown in Fig. 9 whereas optimizer was adam which gives better results about the selection of intents. Regarding the dataset, she created her own dataset based on basic chitchat.

```

# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer c
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

```

Fig. 15. Model Structure of Generative Chatbot designed by Aakriti

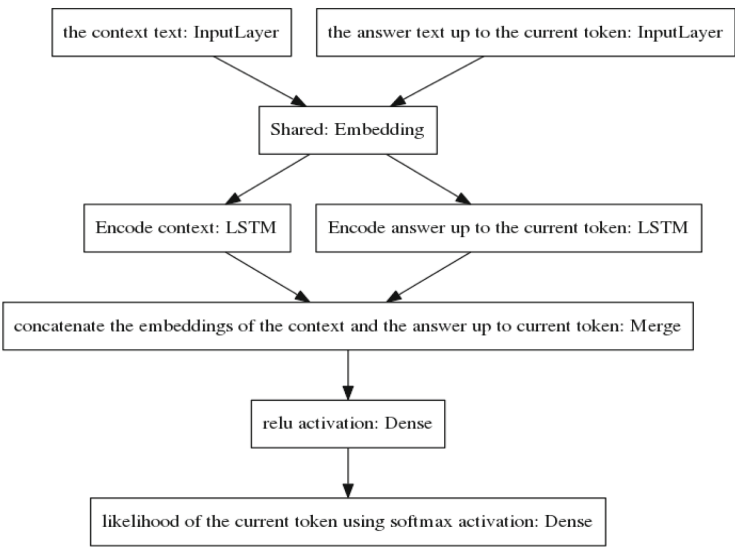


Fig. 16. Workflow of Aakriti’s Model

She followed the steps for model development:

Define Intents: few simple intents and bunch of messages in English Language that corresponds to those intents and also map some responses according to each intent category and created a JSON file named “intents.json” [33].

Data Preparation: then imported nltk, tensorflow, keras and other python libraries. After that loaded the json file and extracted the data. Then used “LabelEncoder()” function provided by scikit-learn which converted the target labels into a model understandable form.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	11392
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 9)	585
Total params: 20,233		
Trainable params: 20,233		
Non-trainable params: 0		

Fig. 17. Aakriti’s Model Neural Network Summary

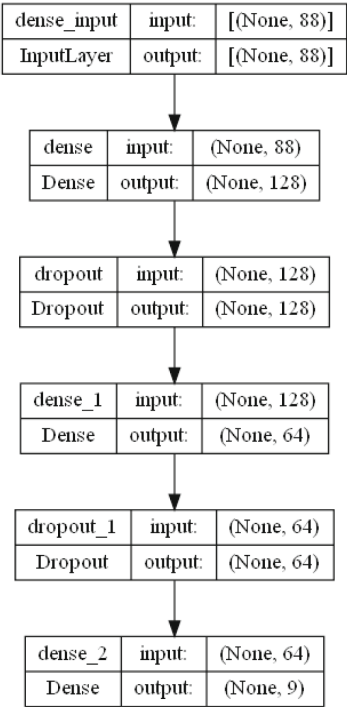


Fig. 18. Aakriti’s Generative Chatbot Model Summary

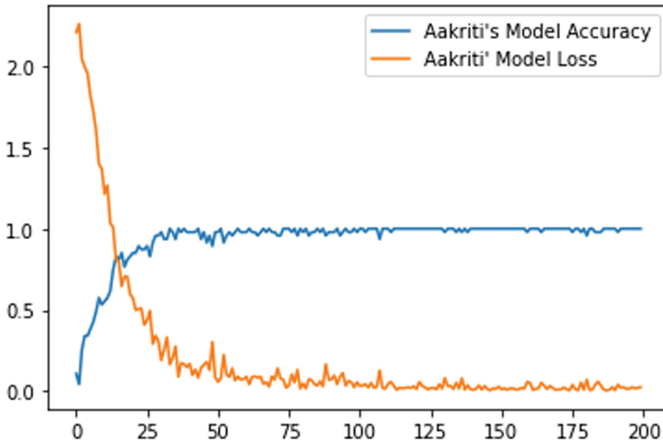


Fig. 19. Aakriti's Generative Chatbot Model Performance Metrics

```
In [6]: # Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

# Compile model. Stochastic gradient descent with Nesterov accelerated gradient
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

#fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5,
model.save('chatbot_model.h5', hist)

print("model created")
```

Fig. 20. Model Structure of Generative Chatbot designed by Shruti

Model Training: Next, she vectorized the text data corpus by using the “Tokenizer” class and it allows us to limit the vocabulary size up to defined number. In further trained the model using “fit” method with training data and labels. We will get more detailed idea after observing the model summary in Fig. 21.

If we observed the Fig. 22 we got idea about the model's accuracy which is approximately 92% and loss 15% showing better performance after 200 epochs. She has used simple dataset but her model is effective for intent based structure for English Language Text Data. This model having drawback about the domain specific question only possible to ask the chatbot.

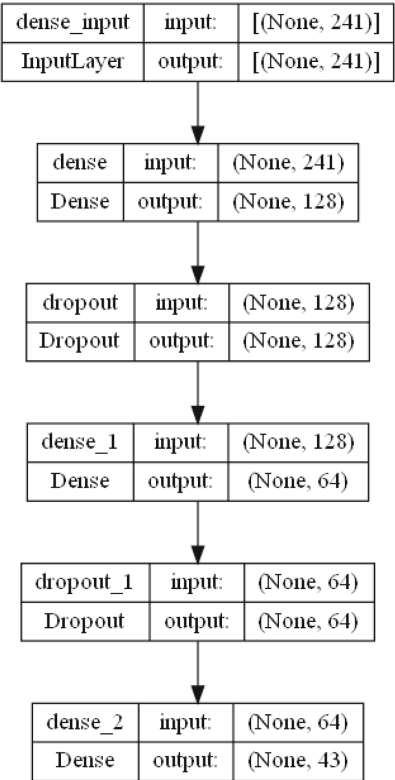


Fig. 21. Shruti Mukherjee’s Seq2Seq Chatbot Model Summary

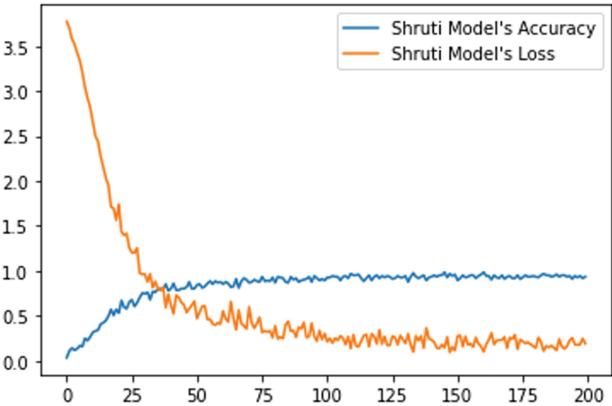


Fig. 22. Shruti Mukherjee’s Seq2Seq Chatbot Model Performance Metrics

```
# Create model - 3 layers. First Layer 128 neurons, second Layer 64 neurons and 3rd output Layer contains
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
```

Fig. 23. Model Structure of Generative Chatbot designed by Tarika

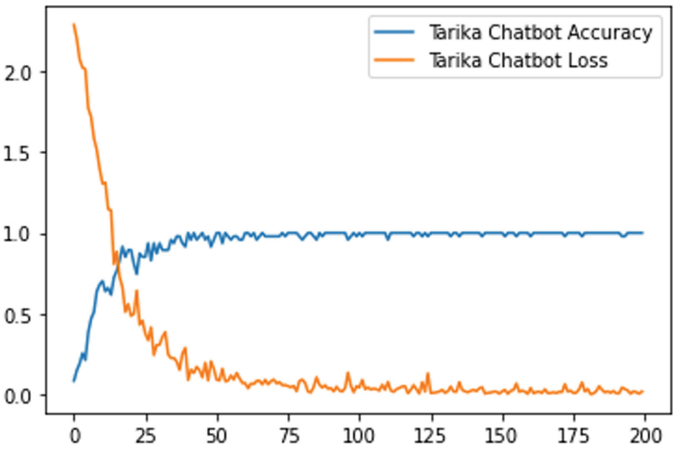


Fig. 24. Tarika Gupta's Seq2Seq Chatbot Model Performance Metrics

3.4 Generative Based Chatbot by Tarika Gupta Weblink: <https://github.com/tarikacs/Generative-based-Chatbot>

The model based on seq2seq and intent pairs dataset designed by Tarika Gupta. The model structure developed by her is as follows (Fig. 23).

In above, Tarika has developed a model using sequential technique where she also has used SGD algorithm for creating neural network. In this neural network, dense network is having 128 layers each and relu function for activation from hidden layer and softmax function for final decision. She also used categorical_crossentropy method for checking the model's loss and accuracy which shown in Fig. 24 whereas optimizer was adam which gives better results about the selection of intents. Detailed idea shown in the Fig. 25 of model summary.

After observation of the Fig. 25 we got idea about the model's accuracy which is approximately 98% and loss 10% showing better performance after 200 epochs.

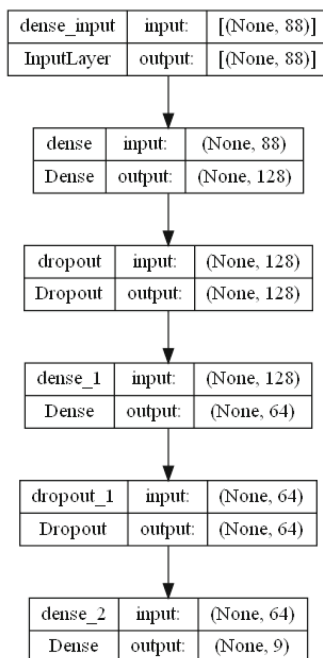


Fig. 25. Tarika Gupta's Seq2Seq Chatbot Model Summary

3.5 Generative Chatbot by Amogh Weblink: <https://github.com/amogh240/GenerativeChatbot>

The model based on seq2seq and intent pairs dataset designed by Amogh. The model structure developed by him is as follows: Amogh has developed a model using sequential technique where she also has used LSTM algorithm for creating neural network. In this neural network, dense network is having 256 layers and softmax function for final decision. In model categorical_crossentropy method used for checking the model's loss and accuracy which shown in Fig. 28 and 29 whereas optimizer was rmsprop which gives better results about the selection of reply. Detailed idea shown in the Fig. 27 of model summary (Fig. 26).

If we observed the Fig. 28 and 29 we got idea about the model's accuracy which is approximately 100% and loss 10% showing better performance after 50 epochs.

In summary, after experimented in python 3 and Jupiter notebook, it was shown that a complete English chatbot model could generate grammatical responses by adding many interactions. In addition, Various Deep Learning techniques like SGD, LSTM, Bi-LSTM and other have own importance to create the good quality chatbot system as a talking product or service. Also, we observed that model's accuracy and loss is not only

```
In [17]: from preprocessing import num_encoder_tokens, num_decoder_tokens, encoder_input_data, \
        decoder_input_data, decoder_target_data, max_encoder_seq_length, max_decoder_seq_length

from keras.layers import Input, LSTM, Dense, Masking
from keras.models import Model
import os

# Choose dimensionality
dimensionality = 256

# Choose the batch size
# and number of epochs:
batch_size = 10
epochs = 50
# Encoder training setup
encoder_inputs = Input(shape=(None, num_encoder_tokens))

encoder_lstm = LSTM(dimensionality, return_state=True)
encoder_outputs, state_hidden, state_cell = encoder_lstm(encoder_inputs)
encoder_states = [state_hidden, state_cell]

# Decoder training setup:
decoder_inputs = Input(shape=(None, num_decoder_tokens))
decoder_lstm = LSTM(dimensionality, return_sequences=True, return_state=True)
decoder_outputs, decoder_state_hidden, decoder_state_cell = decoder_lstm(decoder_inputs, initial_state=
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)

# Building the training model:
training_model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

# Compile the model:
training_model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'], sam

# print("Training the model:\n")
print("Training the model: \n")
# Train the model:
history = training_model.fit([encoder_input_data, decoder_input_data], decoder_target_data, batch_size=
validation_split=0.2)
```

Fig. 26. Model Structure of Generative Chatbot designed by Aakriti

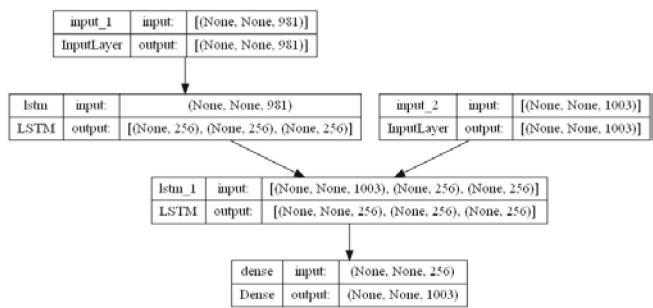


Fig. 27. Amogh's Chatbot Model Summary

depending on Deep Learning Approaches but also on datasets too. Therefore quality dataset is also having its own importance. Finally, we proceed to make the argument that every model is having unique style of python coding and implementation. Based on above examples, we understood current chatbot models in detail for understanding how generating responses and how this affects the quality of the conversation.

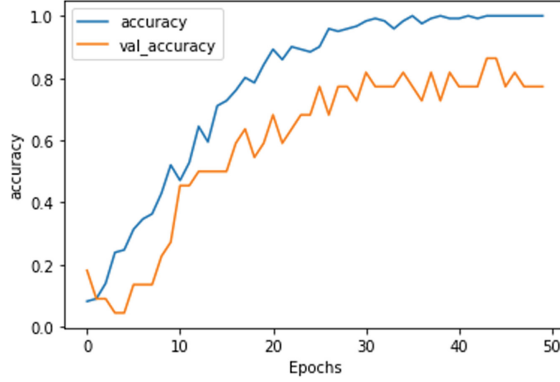


Fig. 28. Amogh's Chatbot Model Performance Metrics: Accuracy

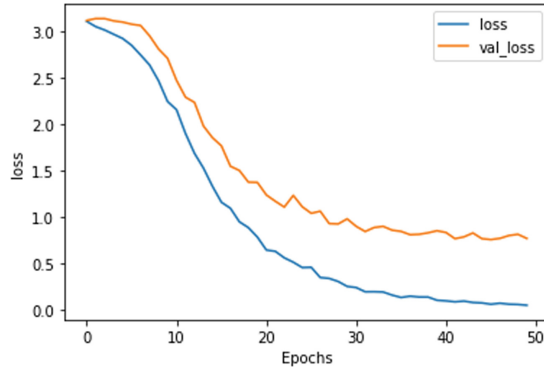


Fig. 29. Amogh's Chatbot Model Performance Metrics: Loss

4 A Proposed Model: MARGEN

Here is the proposed model for Marathi Question Answering Generative Conversation Model coined as “MARGEN” which means who can get system's reply with proper generative answer for the user's query in text and/or audio format. In fact, we already started the source development using IndicBART and Dataset using python 3.0 environment.

MARGEN Conversational System is operationalized through a front-end GUI application, which could be a mobile app or web application, voice assistant and Server side where model will generate the reply. Following steps we proposed in MARGEN model: Firstly, a user or person will ask the question. In next step, User's GUI will send the input query to the system side (Speech to Text format). Now, Input query will get pre-process and in correct tokenization format. After that, the “MARGEN” model gets in the picture, where the question string comes to the model for getting the response as a result where “MARGEN” will understand query's intention and pass it to neural network to accumulate the answer with help of dataset. The trained model will decide the tokenization of the Marathi word, preparing the query in tag format for searching

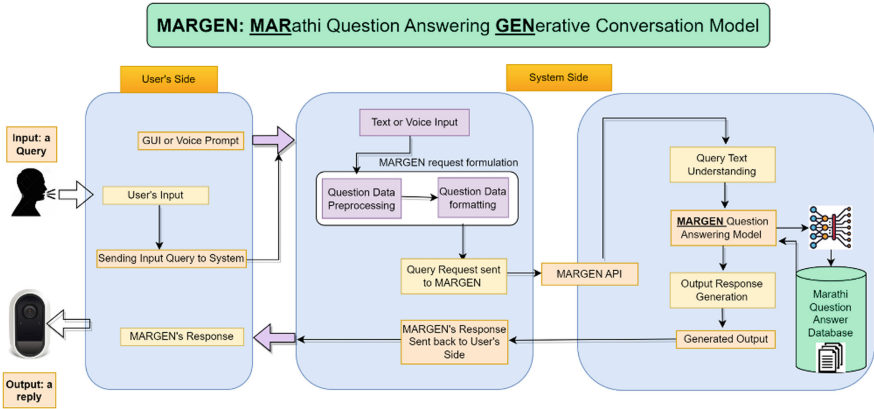


Fig. 30. A Schematic Diagram of Proposed Model: MARGEN

in good response from the dataset defined herewith. The expected answer will generate fetched from dataset and prepared in the text format for further actions. (This is very hard and important step) Finally “MARGEN” will sent the reply to the query asked by the user using the text or audio format (Fig. 30).

5 Conclusion

In the paper, Authors have discussed and claimed that the current research work on Question Answering Model based on English language mostly but very less or none on Marathi Spoken which a challenge for other researchers too. So, Authors proposed the deep learning model along with dataset containing Marathi question answer pairs where it may be useful in cases discussed in motivation part of this paper. In further, the real development of proposed model will get complete in python environment, because Python is Open-Source and having wide scope of predefined libraries which makes easy to developer for developing the real time application. Another challenge is to find the right and appropriate IOT device which can take proper audio from the user and perform the deep learning process in itself to generate output with GPU support for better performance. In addition, Author has to evaluate the model’s performance as a part of quality output. Another article needs to write for result and expected outcome of the same.

References

1. Noga Arikha, Associate Fellow, Warburg Insitute (London): WHAT DO YOU THINK ABOUT MACHINES THAT THINK? <https://www.edge.org/responses/what-do-you-think-about-machines-that-think> (2015)

2. <https://www.globenewswire.com/news-release/2022/06/29/2471371/0/en/Chatbot-Market-Growth-is-projected-to-reach-USD-3-62-Billion-by-2030-growing-at-a-CAGR-of-23-9-Straits-Research.html> (2022).

3. Jang, J.W., Han, W.S. Dialogue system and method for responding to multimodal input using calculated situation adaptability, US Patent 8,719,015 (2014).
4. Chatbot Market: Information by End-User (Large, Medium, Small Enterprises), Business Model (Bot for Service), Type (Standalone, Web-Based), Product Land-scape, and Region — Forecast till 2030 Published on Weblink: <https://straitsresearch.com/report/chatbot-market/toc>
5. Web-Based Technologies Does Your Company Really Need a Chatbot? by P.V. Kannan and Josh Bernoff May 21, Harvard Business Review (2019).
6. T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, \Huggingface’s transformers: State-of-the-art natural language processing,” ArXiv, vol. abs/1910.03771, (2019).
7. Sordoni, A.; Bengio, Y.; Vahabi, H.; Lioma, C.; Grue Simonsen, J.; Nie, J.Y. A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion. In Proceedings of the CIKM ’15 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, Volume 19, pp. 553–562, (2015).
8. Li, J.; Monroe, W.; Ritter, A.; Galley, M.; Jianfeng, G.; Jurafsky, D. Deep Reinforcement Learning for Dialogue Generation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, pp. 1192–1202, (2016).
9. Feng, S., Chen, H., Li, K., Yin, D., Posterior-gan: towards informative and coherent response generation with posterior generative adversarial network. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 7708–7715, (2020).
10. Gao, J., Galley, M., Li, L., Neural approaches to conversational AI: question answering, task-oriented dialogues and social chatbots. In: Now Foundations and Trends.
11. Huang, M., Zhu, X., Gao, J., 2020. Challenges in building intelligent open-domain dialog systems. ACM Trans. Inf. Syst. 38, 1–32. (2019b).
12. Kim, B., Ahn, J., Kim, G., Sequential Latent Knowledge Selection for Knowledge- Grounded Dialogue, 07510 arXiv preprint arXiv:2002, (2020).
13. Li, L., Xu, C., Wu, W., Zhao, Y., Zhao, X., Tao, C., Zero-resource Knowledge- Grounded Dialogue Generation arXiv preprint [arXiv:2008.12918](https://arxiv.org/abs/2008.12918), (2020).
14. Huster, K., Smith, E.M., Ju, D., Weston, J., Multi-modal Open-Domain Dialogue arXiv preprint [arXiv:2010.01082](https://arxiv.org/abs/2010.01082), (2020).
15. Song, Y., Yan, R., Li, C.T., Nie, J.Y., Zhang, M., Zhao, D., An Ensemble of Retrieval- Based and Generation Based Human-Computer Conversation Systems. IJCAI, pp. 4382–4388, (2018).
16. Tao, C., Mou, L., Zhao, D., Yan, R., Ruber: an unsupervised method for automatic evaluation of open-domain dialog systems. In: Thirty-Second AAAI Conference on Artificial Intelligence, (2018).
17. Wang, Y., Liu, C., Huang, M., Nie, L., Learning to ask questions in open-domain conversational systems with typed decoders. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Vol 1, pp. 2193–2203, (2018).
18. Priya Shree The Journey of Open AI GPT models <https://medium.com/walmartglobaltech/the-journey-of-open-ai-gpt-models-32d95b7b7fb2>, (2018).
19. Adaku Uchendu, Zeyu Ma, Thai Le, Rui Zhang, Dongwon Lee, TURINGBENCH: A Benchmark Environment for Turing Test in the Age of Neural Text Generation [arXiv:2109.13296](https://arxiv.org/abs/2109.13296) [cs.CL] <https://doi.org/10.48550/arXiv.2109.13296>, (2021).
20. D. Yin, L. Dong, H. Cheng, X. Liu, K. Chang, F. Wei and J. Gao “A Survey of Knowledge-Intensive NLP with Pre-Trained Language Models”, Computation and Language, arxiv <https://doi.org/10.48550/arXiv.2202.08772>, (2022).
21. I. Beltagy, K. Lo and A. Cohan “ SciBERT: A Pretrained Language Model for Scientific Text”. Computation and Language, <https://doi.org/10.48550/arXiv.1903.10676>, (2019).

22. Márk Lajkó, Dániel Horváth, Viktor Csuvik & László Vidács, Fine-Tuning GPT-2 to Patch Programs, Is It Worth It? ICCSA 2022: Computational Science and Its Applications – ICCSA 2022 Workshops pp 79–91 (2022).
23. Värtinen, Susanna Generating Role-Playing Game Quest Descriptions With the GPT-2 Language Model Roolipelitehtävien kuvausten generointi GPT-2-kielimallilla (2022).
24. Deepali Bajaj, Anita Goel, S. C. Gupta & Hunar Batra MUCE: a multilingual use case model extractor using GPT-3 International Journal of Information Technology volume 14, pages 1543–1554 (2022).
25. Floridi, L., Chiriatti, M. “GPT-3: Its Nature, Scope, Limits, and Consequences”. *Minds & Machines* 30, 681–694. <https://doi.org/https://doi.org/10.1007/s11023-020-09548-1>, (2020).
26. O. Tafjord and P. Clark “General-Purpose Question-Answering with Macaw”. *Computation and Language*, <https://doi.org/10.48550/arXiv.2109.02593>, (2021).
27. Roei Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. https://www.mha.gov.in/sites/default/files/EighthSchedule_19052017.pdf In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics, (2019).
28. Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu, Massively multilingual neural machine translation in the wild: Findings and challenges. CoRR, abs/1907.05019, (2019).
29. Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov, Unsupervised cross-lingual representation learning at scale. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8440–8451, Online. Association for Computational Linguistics, (2020).
30. Raj Dabre and Atsushi Fujita. Combining sequence distillation and transfer learning for efficient low-resource neural machine translation models. In Proceedings of the Fifth Conference on Machine Translation, pages 492–502, Online. Association for Computational Linguistics, (2020).
31. Tejas Dhamecha, Rudra Murthy, Samarth Bharadwaj, Karthik Sankaranarayanan, and Pushpak Bhattacharyya, Role of Language Relatedness in Multilingual Fine-tuning of Language Models: A Case Study in Indo-Aryan Languages. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 8584–8595, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics, (2021).
32. Vikrant Goyal, Anoop Kunchukuttan, Rahul Kejriwal, Siddharth Jain, and Amit Bhagwat., Contact relatedness can help improve multilingual NMT: Microsoft STCI-MT @ WMT20. In Proceedings of the Fifth Conference on Machine Translation, pages 202–206, Online. Association for Computational Linguistics, (2020a).
33. Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar., XLsum: Large-scale multilingual abstractive summarization for 44 languages. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4693–4703, Online. Association for Computational Linguistics, (2021).
34. Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar, IndicNLP Suite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4948–4961, Online. Association for Computational Linguistics, (2020).

35. Mihir Kale and Abhinav Rastogi, Text-to-text pre-training for data-to-text tasks. In Proceedings of the 13th International Conference on Natural Language Generation, pages 97–102, Dublin, Ireland. Association for Computational Linguistics, (2020).
36. Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, Shruti Gupta, Subhash Chandra Bose Gali, Vish Subramanian, and Partha Talukdar, Murlil: Multilingual representations for indian languages, (2021).
37. Yash Khemchandani, Sarvesh Mehtani, Vaidehi Patil, Abhijeet Awasthi, Partha Talukdar, and Sunita Sarawagi, Exploiting language relatedness for low web-resource language model adaptation: An Indic languages study. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics pages 1312–1323, Online. Association for Computational Linguistics, (2021).
38. Anoop Kunchukuttan, Mitesh Khapra, Gurneet Singh, and Pushpak Bhattacharyya, Leveraging orthographic similarity for multilingual neural transliteration. Transactions of the Association for Computational Linguistics, 6:303–316, (2018).
39. Shashi Narayan, Shay B. Cohen, and Mirella Lapata, Don't give me the details, just the summary topic-aware convolutional neural networks for extreme summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics, (2018).
40. A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, Improving language understanding by generative pre-training,” URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf (2018).
41. Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for the squad. In Proceedings of the Association for Computational Linguistics (2018).
42. Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In Proceedings of Empirical Methods in Natural Language Processing (2016).
43. Zhou, X., et al.: Recurrent convolutional neural network for answer selection in community question answering. Neurocomputing. 274, 8–18 (2018).
44. Roy, P.K., Singh, J.P.: Predicting closed questions on community question answering sites using convolutional neural network. Neural Comput. Appl. 32, 10555–10572, (2020).
45. Lukovnikov, D., Fischer, A., Lehmann, J.: Pretrained transformers for simple question answering over knowledge graphs. In: International Semantic Web Conference, pp. 470–486. Springer (2019).
46. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. 9(8), 1735–1780 (1997).
47. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint [arXiv:1508.01991](https://arxiv.org/abs/1508.01991) (2015).
48. Hochreiter, S., & Schmidhuber, J. Long short-term memory. Neural Computation, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>, (1997).
49. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), (2019).

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

