Implementing Natural Language Generation Through Industry-Specific Chatbots

Ankit Kapoor[1], Sujala Shetty[2]

BITS Pilani Dubai Campus, UAE

Corresponding author: Ankit Kapoor, Email: ankitkapoor0406.ak@gmail.com

Natural Language Generation (NLG) is at the forefront of research in the field of Natural Language Processing (NLP) where an algorithm is taught to create text in any given language. The purpose of this research paper is to implement NLG algorithms into the design of a conventional chatbot to improve its design and make it more usable. The paper explores the details and flaws of widely used chatbot templates and aims at providing a way to resolve these issues by implementing a text generation algorithm. The task is performed by assessing an industry-specific chatbot- in this case, a medical query bot- which runs on the proposed principle. The paper also showcases an array of conversational AI's crafted using a publicaly available version of GPT 2, to further explore the scope of Natural Language Generation.

Keywords: Natural Language Processing (NLP), Natural Language Generation (NLG), Language Models, Recurrent Neural Networks (RNN), LSTM, Transformer Models, GPT 2, Language Generation, Text Prediction, Text Classification, Artificial Intelligence, Machine Learning, Deep Learning

2023. In Satyasai Jagannath Nanda & Rajendra Prasad Yadav (eds.), *Data Science and Intelligent Computing Techniques*, 55–67. Computing & Intelligent Systems, SCRS, India. https://doi.org/10.56155/978-81-955020-2-8-6

1 Introduction

Natural Language Generation (NLG) is the forefront of research in the field of Natural Language Processing (NLP) where an algorithm is taught to create text in any given language. The purpose of this research paper is to implement NLG algorithms into the design of a conventional chatbot to improve on its design and make it more usable. The paper explores the details and flaws of widely used chatbot templates and aims at providing a way to resolve these issues by implementing a text generation algorithm. The task is performed by assessing an industry-specific chatbot- in this case, a medical query bot- which runs on the proposed principle. [1] The paper also showcases an array of conversational AI's crafted using publically a publically available version of OpenAi's language model, GPT 2, to further explore the scope of Natural Language Generation.

1.1 Overview of chatbots

Chatbots have been used professionally as a means to replace the human grunt work when it comes to conversational redundancies. Any reputed website or organization has some form a chatbot deployed on their website as an extra step of immersion to get users accustomed to what the organization has to offer. It is an extremely useful tool when used correctly. Fields like Marketing, Supporting Systems, Education, Health Care, Cultural Heritage, and Entertainment use chatbots regularly. [1]

The basic categories of chatbots used professionally are as given in Table. 1

Table 1. Types of chatbots

Knowledge Domain	• Generic • Open Domain • Closed Domain
Service Provided	• Interpersonal • Inter-agent
Goals	• Informative • Conversational • Task based
Response Generation Method	• Rule based • Retrieval based • Generative
Human-aid	• Human-mediated • Autonomous
Permissions	• Open-source • Commercial

1.2 Drawbacks in conventional chatbots

The conventional chatbot follows a simple structure: it consists of an input function that cleans up and inputs the prompt given to it by a human user. Upon receiving the input an engine assesses the text to try and identify the intent of the user's question, that is, it tries to figure out what the user is trying to ask. [2] Once the bot is certain about the intents, it searches through a file with prefab replies to carry forward the conversation. A typical reply file looks something like this:

```
{"tag": "greetings",
    "patterns": ["Hi there", "Is anyone there?", "Hey", "Good morning", "Good evening", "Hi"],
    "responses": ["Hi!", "Hi! Good to see you again", "Hello there", "Hello from the other side haha", "Hello,
human"],
    "context": [""]
    }
```

The chatbot further grabs these prefabs and sends them as replies to the user to carry forward the conversation. This process keeps repeating in a loop for as long as the user wants to carry on the conversation.

This method while accepted has a few drawbacks. Since the chatbot uses a template, there is very limited usability regarding the conversations it can have. It becomes a hyper-specific reply scenario that falls apart when asked an unforeseen question. The accuracy of such a chatbot also falls when users ask it unexpected questions because replies for those questions may not be present in the templates. Lastly, to make such a bot accessible over a wider scope, the developers may have to program a large number of such prefabs, resulting in inefficient use of space and time. [2, 11]

1.3 Natural Language Generation in chatbots

Natural Language Generation aims to solve these issues. The process of text generation eliminates the need to have prefabricated replies. By the use of language models, developers can cut down on.

2 Dataset Description

2.1 Text Generation Dataset

The dataset used for text generation through the use of RNNs is a personally compiled dataset of poems. The dataset consists of 38 poems, all personally composed. The aforementioned dataset is created such that it splits the poems into individual data points such that an individual poem is an individual data point.

Since the text generation algorithm uses two different models: RNN using Tensor flow and transformers using GPT 2, the same dataset is formatted differently for both exercises. For the former, the dataset is in the form of a single text file. The algorithm reads the data in its raw text format and uses it for further text generation. The raw text is provided structure in the given format:

“The closest I've come
To unshroud your visage
Is that you exist
Probably because I think you do...”

In the case of use for GPT 2, the dataset follows a format where every single data point is split into two halves: “Prompt” and “Generated”, and one data point is separated from another by inserting a literal at the end of every poem. The following format is implemented:

Prompt: “The closest I've come\nTo unshroud your visage\nIs that ...”
Generated: “I don't see how you fit\nWithin quarks and electrons ...”
<|endoftext|>

2.2 General Conversation Dataset

The dataset used for training GPT 2 on general conversations was provided by the Amazon Alexa Topical-Chat repository which consists of human-on-human open domain conversations. The aforementioned dataset was made available as open source by Amazon as a part of a language model contest in 2019.

The dataset contains context-specific conversations between two humans. The original dataset is a collection of multiple bursts of small conversations based around a single context (articles or topical news). The conversations are initially structured as a JSON file. This JSON format had multiple redundant fields and hence had to be cleaned to fit the input requirements of GPT 2. Hence, upon formatting the dataset and eliminating all redundant information, the remaining data had the following format:

User: hey hows it going, do you know what the first cloned animal was?
Bot: Was it a sheep? I heard the first cloned pet was a cat, though. Its name was copycat. That is funny!
<|endoftext|>

2.3 Medical Queries Dataset

For the industry-specific medical chatbot, medical queries from multiple online resources were scraped and used. For the present dataset, four specific resources: e-health forum, iCliniq, question

Doctor, and WebMD. The following resources have publically available questions and answers regarding multiple medical topics. The raw JSON dataset has roughly fifty thousand pairs of questions and answers where the original websites for the questions are linked in the "url" field, along with relevant tags which pertain to the topic of the question. [3]

To fit the following data for GPT 2, the dataset was formatted to only contain input and output prompts in the form of questions and answers. The following is the final data on which GPT 2 is trained to answer medical queries:

> Question: how do i stop smoking now
>
> Answer: stopping smoking is about will power and being steadfast. you can stop safely by having bupropion or nicotine patch cover initially in consult with a doctor. contact an addiction clinic near you. wishing you best of health thanks <|endoftext|>

3 Methodology

3.1 Text Generation Methods

The basic idea behind a word-level text-generating algorithm is very similar to the idea of a word prediction model. Here, the algorithm tries to predict the next word in the sentence based on the words which are already present. [5] To further formulate this idea, let us assume that a sentence S is composed by N words in the following way:

$$S = (w_1, w_2, w_3, \ldots, w_N) \quad (1)$$

Each word, "w", while being in the sentence is a part of a larger collection of words known as the "vocabulary", V:

$$V = \{v_1, v_2, v_3, \ldots, v_{|V|}\} \quad (2)$$

Where "|V|" is the size of the total vocabulary. With this given data, the goal of the text generation algorithm is to determine the probability of any given sentence "S" with "W_N" words. So can be done by applying the chain rule as follows:

$$p(S) = p(w_1, w_2, \ldots, w_N) = p(w_1) \cdot p(w_2|w_1) \cdot p(w_3|w_2, w_1) \ldots \cdot p(w_N|w_{N-1}, w_{N-2}, \ldots, w_1) \quad (3)$$

Or,

$$p(S) = \prod_{n=1}^{N} p(w_n|w_{<n}) \quad (4)$$

Here, "$w_{<n}$" refers to the collection of words preceding the time t. The probability of the word occurring at instance t depends on the words previously present in the sentence. Extrapolating this idea, we can determine the probability of a sentence by determining the probabilities of all individual words in the sentence. A language model aims to do this by predicting the word w_{n+1} based on all words $w_{<n}$. [5, 11]

This form of Natural Language Generation can be achieved through multiple machine learning algorithms. For this research paper, the two focused algorithms are Recurrent Neural Networks (RNNs), and Transformers (using GPT 2).

3.1.1. Implementing RNN

As humans, we tend to base most of our understanding around context. For instance, the sentence "Rajasthan is the largest state of _____" is particularly easy to complete with the word "India". The reason is, humans perceive the context from pre-existing words in the sentence to reason out the missing word. [5, 11]

The idea behind the architecture of Recurrent Neural Networks is quite similar: exploitation of data in a sequence. The term "Recurrent" suggests that the same operation is performed repeatedly on every element in a feedback loop where the output depends on previous operations and current input. [4]

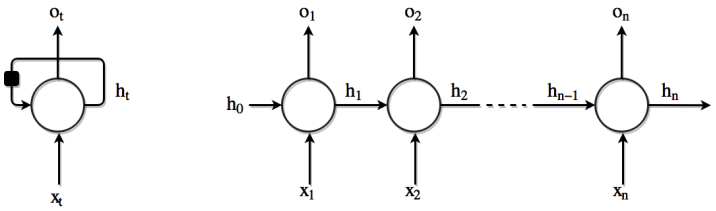

Fig. 1: Folded and unfolded computational diagram of a single instance in an RNN

The architecture depicted in Fig. 1. can be used to model language. For instance, the likelihood of the sentence "This is a new dog" needs to be calculated. We can do so by applying the chain rule as follows:

$$p(This), p(is|This), p(a|This\ is), \dots, p(dog|This\ is\ a\ new) \quad (5)$$

The next step is to define how these values are input to the language model. The input sequence of size N-1 words is input to the model through the application of 1-of-K encoding, or one-hot-encoding scheme. Every word in the vocabulary V is assigned a vector in binary, and every other symbol in the sequence is assigned 0:

$$w_i = [0,0,\dots,1(i^{th}\ element),0,\dots,0]^N \in \{0,1\}^{|V|} \quad (6)$$

Upon one-hot-encoding all symbols in a sentence, each vector is multiplied by a E (weight matrix) to attain continuous vectors $(x_1, x_2, x_3, \dots, x_{N-1})$ such that $x_j = E^N w_j$.

The next step would be to input this data into the RNN. The following figure illustrates this process:

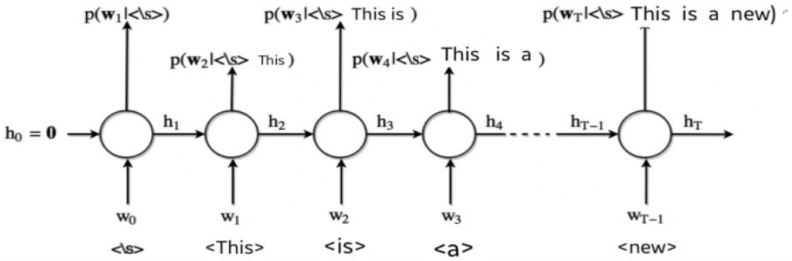

Fig. 2: RNN for language model

As demonstrated in Fig. 2., memory vector, h, is initialized and a special token is provided to it to mark the beginning of the sentence. The output is the probability of every single word. At each step, the memory vector gets updated and passed on to the next step. [4] This process is repeated for the following step, where now the input is the word "This" and $p(w_2|<\backslash s> This)$ is the output.

At each step, a probability distribution for every word in vocabulary V is calculated. RNN outputs a

softmax layer returning a $|V|$ sized vector such that its i^{th} element produces the predicted probability of the word V_i. This can be calculated as follows:

$$p(w_n = k|w_{<n}) = \frac{\exp(V_k^N h_t + b_k)}{\sum_{k'} \exp(V_{k'}^N h_t + b_{k'})} \quad (7)$$

Now that the algorithm can produce sound results for scoring sequences using RNNs, it needs a loss function to assist it in the process of learning. The loss function is the negative log probability that the model assigns to the correct output. This can be calculated by $L = -\log(p(w_1, w_2, w_3, \ldots, w_N))$. Hence, for a sequence "x":

$$L(x) = -\sum_n \log_{Pmodel}(w_n = x_{n+1}) = -\sum_n \log o_n[x_{n+1}] \quad (8)$$

Where $o_n[x_{n+1}]$ is the corresponding element of the softmax layer. With the availability of this value, it is now possible to back propagate the loss through previous layers and adjust the weights accordingly.

Generating text using such a language model is now a straightforward task. To generate a new sentence, we initialize a context vector randomly, let the RNN sample the text at each step, and let it feed one word back into the input from the output. Repeating this process for a set number of words allows us to generate new text. [4]

3.1.2. Implementing Transformers using GPT 2

GPT 2 (Generative Pretrained Transformer) is one of the most advanced text generation models made available by OpenAI. The model is a text generator trained over roughly forty gigabytes of text data scraped from all over the internet. GPT 2 by itself is available in multiple sizes and complexities depending on the quality of training. For this research, the smallest publically available iteration of GPT 2, namely, distill GPT 2 was used because of operational constraints. This model was made available through the HuggingFace transformer library.

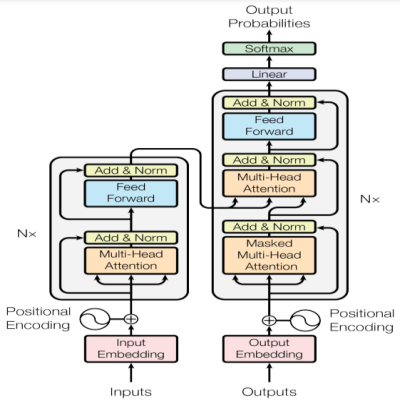

Fig. 3: Basic architecture of the transformer language model

GPT 2 famously uses the transformer model depicted in Fig. 3. for text generation and classification. A vast majority of competitive neural sequence transduction models follow the encoder-decoder architecture. Essentially an input sequence of symbol representations $(x_1\, x_1, \ldots, x_n)$ is mapped by the encoder to a continuous representation z = $(z_1\, z_1, \ldots, z_n)$. Upon receiving z, the decoder generates a sequential output series $(y_1\, y_1, \ldots, y_n)$ one element at a time.

The model is auto-regressive, meaning that at each step it takes the output of the previous step as an additional input for generating text at the current layer. The aforementioned figure describes the basic architecture of the transformer model. The model uses stacked self-attention and point-wise fully connected layers of encoders and decoders. [6]

The encoder comprises six identical layers. Each layer has two sub-layers: a multio-head self-attention mechanism and a position-wise connected feed-forward network. A residual connection is employed around each layer which is followed by a layer normalization. The output of each layer is depicted as $LayerNorm(x + Sublayer(x))$ where $Sublayer(x)$ is the function that the sub-layer implements on itself. Similar to the encoder, the decoder is comprised of six identical layers. Each layer, in addition to the two sublayers present in the encoder, has s third sublayer which performs multi-head attention over the output of the whole stack. There is a residual connection between each sub-layer preceding the normalization layer. The self-attention layers in the decoder are modified to prevent positions from attending to subsequent positions. The outputs from these layers are offset by one position, hence ensuring that the output for the layer I only depends on known outputs from $<i$ layers. [6, 8]

The attention layer can be described as a vector mapping function where multiple vectors, namely, queries, keys, values, and outputs are all mapped as a set of queries and key-value pairs. The output is computed as a weighted sum of these values, where the weights are calculated by calculating the probability of each query with each key. [6]

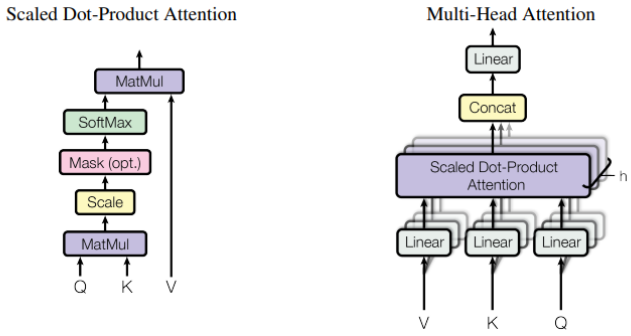

Fig. 4: (L) scaled dot-product attention, (R) multi-head attention across all sublayers

The attention function used by the encoder and decoder layers is called "Scaled Dot-Product Attention". The input is comprised of queries and keys, all of dimension d_k. In practice the computation of the attention function happens over a set of queries simultaneously in a matrix, Q. Keys and values are stored in matrices K and V respectively as visualized in Fig. 4. The output is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (9)$$

Instead of performing a single attention function with d_{model} dimensional keys, values, and queries, it is beneficial to linearly project these values h times with learned projections to d_k and d_v dimensions respectively. On each of these projected values, the attention function is performed in parallel yielding d_v dimensional output values. The aforementioned values are projected and concatenated again to output the final values. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this. [6. 8]

$$MultiHead(Q,K,V) = Concat(head_1, head_2, \dots, head_h)W^O \quad (10)$$

$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (11)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_k}$, and $W_i^o \in \mathbb{R}^{d_{model} \times d_k}$. Here $h = 8$ parallel attention layers are employed. For each layer $d_k = d_v = d_{model}/h = 64$. The net computational cost is similar to that of single–head attention with full dimensionality.

The model utilizes multi-head attention in three specific ways: (1) the queries come from the previous decoder layer, and memory and key values come from the output of the encoder in the encoder-decoder attention layer. Hence all positions in the decoder are allowed to attend to overall positions in the input sequence. (2) the encoder contains self-attention layers. In self-attention layers. In each of these layers' keys, values, and queries come from the output of the previous layer. Each position in the encoder can attend to all positions in the previous layers of the encoder. (3) The same applies to all positions in the decoder layers as well. To prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. [6, 8]

3.2 Text Generation in Chatbots

This research aims to replace the conventional chatbot model with something faster by implementing natural language generation. For this purpose, the distill GPT distill has been used as the primary text generation model. A regular chatbot would use a file containing pre-set replies to hold a conversation with the user and would eventually lose accuracy as soon as the questions become more and more complicated. This is where text generation comes in: we replace the conventional text file with a text generation engine to maintain a conversation for longer in a manner where it feels like the user is conversing with a real person on the other end. [7]

This idea has been implemented in two separate chatbots: Villager- a general purpose and MedBot- a chatbot for industry-specific use (medical queries).

3.2.1 General Conversation Chatbot (Villager)

Villager is coded in python and relies on Distil GPT 2 made available by HuggingFace. It has been trained on a set of roughly fifty thousand queries extracted from the Topical-Chat conversations made available by Amazon. A detailed explanation of how it was coded can be found here (https://github.com/ankitkapooor/Villager/blob/main/README.md). [9]

3.2.2 Medical Chatbot (MedBot)

MedBot takes the idea applied in the previous chatbot and extends it to an industry-specific chatbot, in this instance, a medical bot. the idea of MedBot is to become a helpful aid to anyone having a question concerning the medical field. It is trained on a dataset compiling questions and answers available on multiple internet sources such as WebMD. A detailed explanation of how it was coded can be found (https://github.com/ankitkapooor/MedBot/blob/main/README.md). [10]

3.3 Text Summarization

One pattern which emerges when we take a closer look at MedBot and the way it converses with the user, we observe that a lot of the replies generated by the algorithm are anecdotal, such that they are formed in terms of personal experiences. A major reason for this is that the training data is gathered from multiple online sources which consist of anecdotal question-answers, hence making the algorithm generate text in a similar way. This dilutes the necessary information being covered in the answer and makes it less accurate for the user. One projected way to tackle this is by running

a text summarization algorithm on the generated answers to rid them of any redundant information, hence leaving only the necessary parts.

3.3.1 Implementation in MedBot and Villager

Text summarization mainly falls under two categories, abstractive text summarization, and extractive text summarization. Abstractive summarization involves more work and resource intensity as it is a vastly complicated process. It includes understanding the given text and listing out all the information in a form such that no information is lost while also contracting the overall size of the text body. Tasks like simplifying metaphors or clubbing points together to make a more efficient argument, etc. are all a part of abstractive summarization. Extractive summarization on the other hand is a simpler process. It involves reading the given text body and extracting only useful words, phrases and sentences from it to come up with a summary that preserves the essence of the original text. [13]

A process like extractive summarization has the potential to eliminate anecdotal redundancies from the answers generated by MedBot while also keeping specific words, phrases, and sentences that make up the necessary answer.

Hugging face has a default summarization that performs extractive summarization on text. It uses another language model, BERT for this purpose. This summarization model is comprised of a series of BERT decoders and summarization classifiers. The BERT encoders are pre-trained on the task of text summarization. Extractive summarization behaves like a classification problem at the level of a sentence. Essentially, the algorithm labels each sentence $y_i \in \{0, 1\}$ such that it decides whether or not the sentence should be included in the final summary. Having a start token and an end token differentiates the sentences. Upon attaining a vector for each sentence, a feed-forward layer is used to score each sentence based on relevance. This relevance metric scores words in a sentence on word popularity throughout the text body, hence eliminating less useful information. The basic architecture can be observed in Fig. 7. [14]

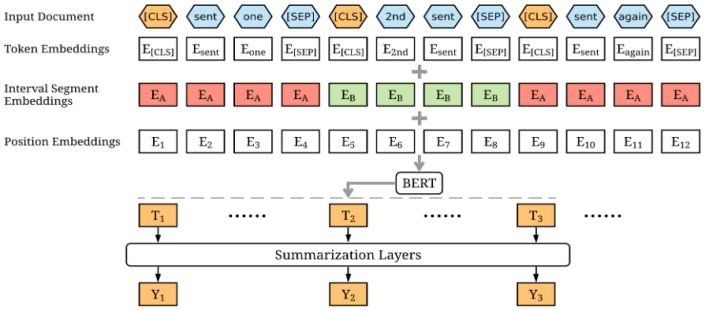

Fig. 7: BERT summarization architecture

Furthermore, the use of an intelligent summarization algorithm opens the door to making a general-purpose conversational bot like Villager more useful. A chatbot is only as accurate as the data it has been trained on. For training Villager, a dataset made available by Amazon in 2019 was used. This effectively limits the topical intelligence of the bot to the events of 2019. This means that the bot would render inaccurate replies to any topical questions, such as "Who is the current President of the USA?"

Adding a pipeline that can differentiate topical questions from conversational questions can tackle this problem. Upon identifying a question as topical, it can be passed to an algorithm that scrapes

the web for relevant information around said question, summarizes it effectively, and returns a reply in the form of an answer by the bot.

3.3.2 Drawbacks

Applying a summarization filter to the generated replies did help in coming up with more relevant information as it eliminates redundant parts. In doing so, even though the answers seem more precise there is a very high time delay between the user asking the question and the bot replying.

```
PS C:\Thesis> python acc2.py
2022-06-19 13:02:35.809187: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic libra
ry 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-06-19 13:02:35.809607: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do n
ot have a GPU set up on your machine.
Without Summarization: --- 6.881376504898071 seconds ---
With summarization: --- 130.12976861000006 seconds ---
```

Fig. 8. MedBot outputs with text summarization

There is a very high increase in time delay when the answers are passed through the summarization pipeline as shown in Fig. 8. Moreover, extractive summarization while being a useful tool to summarize text to preserve the essence of the original piece also leads to loss of information in case of larger text bodies.

4 Results

4.1 Traditional Methods and Shortcomings

Natural Language Generation delves into creating text in human language, which can be very ambiguous. Generating text is a very open-ended task and for that reason, there is an ongoing debate regarding the kind of methods put to use to attain a relevant accuracy measure. Unlike working with numbers, the often overlooked fact is that more than one kind of answer can be correct when it comes to language tasks. For example, a question like “Is football your favourite sport?” can have an answer “Yes, I love playing football.” The statement “No, I enjoy f1 more than I enjoy football” can also be interpreted as an answer to the question. In no way are the two answers similar: no bag of words combination or entity extraction will result in both the answers being similar to each other. Yet, both are equally correct when seen in the context of the question asked. Traditional NLP methods which rely on similarity-based criteria often overlook this fact and end up providing a figure which is inaccurate in more than one way. [15]

To test the language model by traditional methods, a python library called “Rouge” was used. Rouge stands for “Recall-Oriented Understudy for Gisting Evaluation. It is a common tool used to evaluate NLP engines. It focuses on recall using ROUGE-N, ROUGE-S, AND ROUGE-L. R-N is a measure if $n \in \{1, 2, 3\}$-grams, R-L is used to measure the longest matching text sequence. R-S takes into consideration any pair of words in the sentence in order, which permits arbitrary gaps. This is also known as skip-bigram. [16]

To measure accuracy using this method, the amazon conversation dataset was split into training and validation sets, with a little over 700 conversations in the validation set, never before seen by the language model. Plugging the language model into these questions, answers for all of them were generated and noted down. Upon having a complete set of references (answers in the validation dataset) and predictions (answers generated by the chatbot), the rouge accuracy test was run on the model with the following results:

It tabulated the recall at 25.14%, precision at 42.85%, and the f1 score at 32.14%. These figures refer to the recall and precision of the generated answer in context with the answer stored in the validation dataset. They only provide a metric of how similar the two answers are and overlook the far more important metric of deciding whether the answers are both valid or not. Hence, for the purpose of our paper, we have chosen to test the model against human intelligence.

4.2 Testing Against Human Intelligence

It makes sense to involve a human component in testing a language model like ours. Unlike

traditional testing methods which can't determine the validity of a response to a question, humans are extremely good at that skill because that is the nature of how a human conversation takes place. To get an agreeable performance figure, a test was set up where human subjects were presented with ten pairs of conversations: one between me and another human, and the other between me and the language model. [17] The humans I had a conversation were not aware of the exercise and were told about the purpose of the exercise once the conversation ended. Upon reading the conversations, the human subjects were asked to pick out the conversation between me and the language model. The questionnaire with all the conversations can be found here (https://forms.gle/HirvEUFWekbqK7ox8)

Roughly sixty people took the accuracy test, providing 580 data points to calculate the average scores. At an average, roughly 44.31% times was a person able to call out the AI. The results are graphed in Fig. 9.

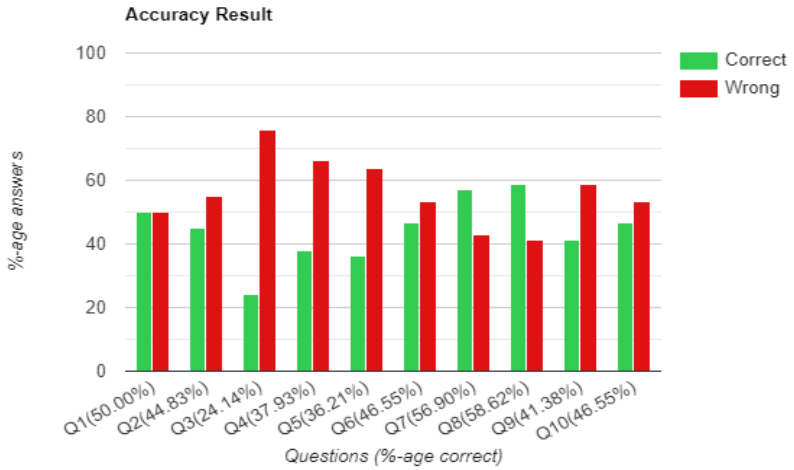

Fig. 9: Per question accuracy scores graphed

Upon closer examination, it is visible that in only two out of the ten provided sets does the green bar cross the red: the other 8 times the conversational AI is able to successfully pass off as human. The traditional method of accuracy overlooked an important aspect of validity which is much better captured by the human test.

5 Conclusion and Future Scope

Through the course of this research, we have looked at the scope of Natural Language Generation and explored its industrial applications. Right from the shortcomings of older, template chatbots to cutting-edge chatbots that implement NLG indistinguishable from human conversation. The exploration was done through multiple different chatbots created during the course of-this work, primarily MedBot and Villager. These two bots apply NLG in both a general conversational sense (in Villager) and in an industry-specific sense (MedBot). Along with these aspects, the paper also explores openly available language models like GPT-2 made available by OpenAI and seeks out ways to personalize and implement them. While the chatbots have acceptable performance, much improvement can be made to them.

Villager experiences a topical handicap when asked about current events, or any topical question for that matter. As showcased earlier, a web-scraping/text-summarization pipeline can help get over this hump. By implementing a classifier that can tell topical questions from conversational questions, the chatbot can rely on extractive summarization to handle things outside the scope of the dataset and still maintain a seamless experience for the user.

When asked whether the bot would mortally harm 1. An old woman, 2. A kid, 3. A black man, and 4. A white man, answered yes to all 4: explaining that even though the bot understood language it didn't understand the ethical implications of its answers. This problem isn't specific to Villager or MedBot; since all such language models are trained on data scraped from the internet, they are prone to perceive stereotypes and biases as normal. A sentiment analysis training shall be done on the AI to make it sensitive to Biases and stereotypes masked in the data it is trained on. [12, 18]

Further, there is an up rise in conversational AIs supplication in fields like therapy and suicide prevention. Such need is popularized by products like Replika, which is a personal AI companion with its conversational AI built on top of GPT-3. [19] It brought about a paradigm shift with how it handles conversations with sensitive users on sensitive topics. It goes to show that Natural Language Generation is a tool that when used properly can quite literally change lives. Through the course of this research, we have scratched the surface of the possibilities in NLG and experienced it first-hand with our chatbots. We strive to work further in this field to create something helpful and available to everyone who may need it.

References

1. Huang, X. (n.d.). Chatbot: Design, architecture, and applications - cis.upenn.edu. Retrieved April 11, 2022, from https://www.cis.upenn.edu/wp-content/uploads/2021/10/Xufei-Huang-thesis.pdf

2. Reiter, E. (1995, April 23). *NLG vs. templates*. arXiv.org. Retrieved April 11, 2022, from https://arxiv.org/abs/cmp-lg/9504013

3. LasseRegin. (n.d.). *Lasseregin/medical-question-answer-data: Medical question and answer dataset gathered from the web*. GitHsub. Retrieved April 11, 2022, from https://github.com/LasseRegin/medical-question-answer-data

4. Ducau, F. (2020, October 30). *Text generation with recurrent neural networks (rnns)*. Paperspace Blog. Retrieved April 11, 2022, from https://blog.paperspace.com/recurrent-neural-networks-part-1-2/

5. Zhang, S., Dinan, E., Urbanek, J., Szlam, A., Kiela, D., & Weston, J. (2018, September 25). *Personalizing dialogue agents: I have a dog, do you have pets too?* arXiv.org. Retrieved April 11, 2022, from https://arxiv.org/abs/1801.07243

6. Alammar, J. (n.d.). *The illustrated GPT-2 (Visualizing Transformer language models)*. The Illustrated GPT-2 (Visualizing Transformer Language Models) – Jay Alammar – Visualizing machine learning one concept at a time. Retrieved April 11, 2022, from https://jalammar.github.io/illustrated-gpt2/

7. Korab, P. (2022, February 18). *Training Neural Networks to create text like a human*. Medium. Retrieved April 11, 2022, from https://towardsdatascience.com/training-neural-networks-to-create-text-like-a-human-23bfdc23c28

8. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, December 6). *Attention is all you need*. arXiv.org. Retrieved April 11, 2022, from https://arxiv.org/abs/1706.03762

9. Kapoor, A. (n.d.). *Villager* . GitHub. Retrieved April 11, 2022, from https://github.com/ankitkapooor/Villager

10. Kapoor, A. (n.d.). *MedBot*. GitHub. Retrieved April 11, 2022, from https://github.com/ankitkapooor/MedBot

11. Wolf, T. (2020, September 2). ☐ *how to build a state-of-the-art conversational AI with transfer-learning*. Medium. Retrieved April 11, 2022, from https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313#79c5

12. *Machine reading comprehension - microsoft AI lab*. Machine Reading Comprehension - Microsoft AI Lab. (n.d.). Retrieved April 11, 2022, from https://www.microsoft.com/en-us/ai/ai-lab-machine-reading

13. Nenkova, A., McKeown, K. (2012). A Survey of Text Summarization Techniques. In: Aggarwal, C., Zhai, C. (eds) Mining Text Data. Springer, Boston, MA https://doi.org/10.1007/978-1-4614-3223-4_3

14. Liu, Y., & Lapata, M. (2019, September 5). Text summarization with pretrained encoders. arXiv.org. Retrieved June 6, 2022, from https://doi.org/10.48550/arXiv.1908.08345

15. Sai, A. B., Mohankumar, A. K., & Khapra, M. M. (2020, October 5). A survey of evaluation metrics used for NLG systems. arXiv.org. Retrieved June 6, 2022, from https://doi.org/10.48550/arXiv.2008.12009

16. Brain, B. G. G., Goodrich, B., Brain, G., Brain, V. R. G., Rao, V., Brain, P. J. L. G., Liu, P. J., Brain, M. S. G., Saleh, M., KenSci, Minnesota, U. of, Corporation, E. V. A., LinkedIn, University, B., & Metrics, O. M. V. A. (2019, July 1). Assessing the factual accuracy of generated text: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM Conferences. Retrieved June 6, 2022, from https://dl.acm.org/doi/abs/10.1145/3292500.3330955

17. Belz, A., & Reiter, E. (2006, April). Comparing automatic and human evaluation of NLG systems. In 11th conference of the european chapter of the association for computational linguistics (pp. 313-320).

18. Barikeri, S., Lauscher, A., Vulić, I., & Glavaš, G. (2021). RedditBias: A real-world resource for bias evaluation and debiasing of conversational language models. arXiv preprint arXiv:2106.03521.

19. Hakim, F. Z. M., Indrayani, L. M., & Amalia, R. M. (2019, February). A dialogic analysis of compliment strategies employed by replika chatbot. In Third International Conference of Arts, Language and Culture (ICALC 2018) (pp. 266-271). Atlantis Press.