SkillBot: Towards Data Augmentation using Transformer language model and linguistic evaluation

Suresh Khatri, Muddesar Iqbal, George Ubakanma, and Spike van der Vliet-Firth
Department of School of Engineering
London South Bank University
London, the United Kingdom
khatris3, m.iqbal, george.ubakanma@lsbu.ac.uk

Jobs and Skills Programme Lead
Economy, Jobs & Partnerships
Lewisham Council

Spike.vanderVliet-Firth@lewisham.gov.uk

***Abstract*— Creating accurate, closed-domain, and machine learning-based chatbots that perform language understanding (intent prediction/detection) and language generation (response generation) requires significant datasets derived from specific knowledge domains. The common challenge in developing a closed-domain chatbot application is the lack of a comprehensive dataset. Such scarcity of the dataset can be complemented by augmenting the dataset with the use of state-of-the-art technologies existing in the field of Natural Language Processing, called 'Transformer Models'. Our applied computing project experimented with a 'Generative Pre-trained Transformer' model, a unidirectional transformer decoder model for augmenting an original dataset limited in size and manually authored. This model uses unidirectional contextual representation i.e., text input is processed from left to right while computing embeddings corresponding to the input sentences. The primary goal of the project was to leverage the potential of a pre-trained transformer-based language model in augmenting an existing, but limited dataset. Additionally, the idea for using the model for text generation and appending the generated embedding to the input embedding supplied was to preserve the intent for the augmented utterances as well as to find a different form of expressions for the same intent which could be expressed by the potential users in the future. Our experiment showed improved performance for understanding language and generation for the chatbot model trained on the augmented dataset indicating that a pre-trained language model can be beneficial for the effective working of natural language-based applications such as a chatbot model trained on the augmented dataset indicating that a pre-trained language model can be beneficial for the effective working of natural language-based applications such as a chatbot.**

Keywords*— *Data Augmentation in NLP, NLG (Natural Language Generation), Natural Language Processing, transformer model, NLU (Natural Language Understanding) Rasa chatbot

I. INTRODUCTION

SkillBot is a closed knowledge, machine learning-based domain virtual agent built using Rasa, an open-source chatbot framework, and deployed for cloud-based information retrieval. Machine learning-based chatbots can be categorized into two categories, intent-detection-based model and generative model. Intent-detection-based chatbots resolve text classification problems where an underlying generative classifier algorithm learns and predicts user intents based on text input. The main purpose of SkillBot software is to test new approaches to engaging and supporting Lambeth, Lewisham, and Southwark Council residents with employment, skills, and career advice. The chatbot can provide information about various topics such as job opportunities, education, career building, and complimentary service, which supports the resident need. In an uncertain UK labour market, this pilot project tests how public services can adapt to deliver high volumes of light-touch and self-directed support, addressing resource challenges for the public sector and information accessibility challenges during a period of significant change in the labour market. Public Services were limited in their ability to deal with the high volumes of support required by residents during the pandemic, from many residents who were not used to accessing support with their employment situation. There is a great deal of employment advice and guidance available online, but it is challenging to navigate and not always good quality advice. Skillbot aims to overcome this navigation challenge by providing verified, good-quality advice based on the challenges residents describe to the chatbot.

A quality dataset is a key ingredient in any machine learning-based project. The machine learning model consumes a dataset for learning and constructing a model with weights that can generalize inputs given to it. The dataset used for training the initial model for the Skillbot chatbot was limited in size. In the case of this project, a scarcity of datasets and the presence of similar utterances across multiple intents and significantly different topics was causing the model to provide unsuitable responses to the user. The model was demonstrating poor NLU accuracy and poor NLG accuracy as the input from the NLU section is received by the NLG section for generating a response. Applying a data augmentation approach to the original dataset with the help of a pre-trained transformer-based language model generated a substantial amount of data for subsequent processing. The chatbot was trained on the original and augmented dataset and tested against a set of 143 different stories. The project then evaluated NLU, NLG, and the overall dialogue management system using a utility provided by the Rasa framework [1]. Both chatbot models were evaluated. The chatbot model trained with an augmented dataset showed much-improved metrics for NLU, NLG, and dialogue management components of the Skillbot chatbot model.

In this paper, we have demonstrated that the potential of a large pre-trained transformer-based language model (e.g., GPT-2 in our case) could be leveraged for improving the language understanding and language generation capability of a chatbot application. One of the applications of those transformer-based models could be augmenting human-generated datasets, creating comprehensive and accurate datasets required for machine learning-based chatbots which perform intent detection as a core NLU task

II. PROBMEL STATEMENT

The effectiveness of a chatbot lies in its ability to identify user intents and message contents correctly and to respond with the appropriate response. Enough utterances for intents and an accurate flow of the stories are required for the training and effective output from the chatbot built using the Rasa framework. [1] supports the fact that a machine learning model needs many examples in the range of thousands to induce function with more generalization capability. The chatbot trained on the original dataset suffered from low accuracy problems for both the NLU and NLG sections. Some of the issues found during the study for the cause of the low-performance metrics are discussed.

A. Limited dataset size

The dataset in the case of the Rasa framework consists of different parts viz. intents, responses, stories, and the domain. An intent represents a category for a user-sent message. It represents an intention coming from a user for knowing or asking something. The chatbot receives intent in the form of voice, text, or quick replies. Intents are used by the Rasa NLU section for training purposes. Each response for individual intents is defined inside the domain file. Each response name begins with the prefix 'utter_' in this case. All responses are defined inside the domain file. Each response can have multiple response texts which are predicted by the response retrieval component in Rasa. The response can also be a function call for execution but in the case of SkillBot custom actions were not used, limited only to text responses. Responses are used by the Rasa Core model for training dialogue policy. Stories represent patterns for conversations that might take place with the user. Stories are defined as a sequence of intents and responses. In the case of SkillBot, most of the stories are defined as a sequence of one intent followed by one response. The implementation details and more theoretical understanding of this jargon are explained here [2]. The initial dataset was authored manually. The dataset consisted of 246 (excluding intents for small talks) different intents. Each of them had around seven short, condensed utterances on average. Most of them were keyword representations of the intents. Each of those intents had their respective response utterance well and stories mainly consisting of one question and one answer. The dataset was balanced but not adequate in size also there were not many variations found among the utterances present for intent. The development of chatbots is suggested to take place using a Conversation Driven Development approach [3]. Since the chatbot was still in the development phase and was not made publicly accessible, following the CDD approach was not possible in our case except for making some changes to the dataset manually.

B. Ambiguous intents

An intent or a category must be unique to each other i.e., an intent must consist of utterances that carry semantics that represent only that category or intent under which it is listed. As the number of intents increases, the possibility remains of creating and listing highly similar intents under multiple intents. This creates confusion and ambiguity for the underlying classification algorithm while classifying an intent for a particular utterance. An NLU test was performed for the Skillbot chatbot model trained with the initial dataset which was carried out using the tool facilitated by the Rasa framework. It was confirmed that intent detection failed for those utterances which showed multiple polarities for the different intents. The presence of utterances carrying bipolar semantics reduced the NLU performance of the Skillbot by creating ambiguity for the classifier algorithm.

C. Lack of Smalltalk conversations training data

The initial dataset did not contain any informal, conversational dataset. The dataset only consisted of domain-specific intents and responses because of which the chatbot would sound less interactive when a user would try to have some more information or personalized conversations. The chatbot response would sound appear less user-friendly and engaging since the dataset for small talk conversation was absent. Facilitating the addition of the Smalltalk dataset to the dataset would help to improve the HCI (Human Computer Interaction) aspect of the chatbot. Manual preparation of an informal conversation dataset would be time-consuming. We incorporated the pre-existing small talk dataset available in the markdown version for the Rasa chatbot from the link mentioned here [4]. The dataset is comprehensive and accurate. The addition of the Smalltalk dataset to our domain dataset increased the number of intents, responses, and stories.

Overall, the issues causing the problem are the small dataset size and inconsistencies found across intents in the dataset.

D. Research Question

Our main focus was on improving NLU and NLG performance for the chatbot. We decided to use the suggested default NLU and NLG algorithms by the Rasa framework for model building as they are considered state-of-the-art technologies in the field of NLP. The challenge presented to us was a smaller dataset. Through this project, we decided to examine if applying dataset augmentation over the existing dataset accompanied by a human reviewing of the augmented dataset for a close knowledge domain chatbot would improve the dialogue management system of the chatbot significantly.

III. LITERATURE SURVEY

The use of transformer-based models has been increased in the field of NLP for data augmentation, especially in case of the unavailability of enough training data resources. The most common scenario present to the developers is the unavailability of a large domain-specific knowledge base for any kind of NLP tasks such as text prediction, classification, and question answering. Irrefutable truth is that machine learning models need a large knowledge base to work with greater accuracy and precision and the same holds with NLP

models. The transformer-based model has received great popularity because of the features such as effective input context representation, and parallelization of tasks achieved by the transformer-based model which was introduced in [5]. The attention mechanism is a key concept in the transformer model. The attention mechanism is inspired by cognitive psychology. [6] suggests that in cognitive psychology the attention mechanism works in two layers. In the first step parallel processing happens and attention is provided uniformly to all the information available. In the second step, focus or attention is made to a special part of the information available.

The GPT2 model was introduced by [1]. The GPT2 model is a transformer-based language model (LM) consisting of only decoders created by OpenAI for text generation. There are different variants of GPT-2 models based on the dataset it is trained on and the number of parameters it has. The model we used for our purpose is the GPT-2 large model. GPT-2 is a large-scale transformer model that is pre-trained on WebText(40GB), has a vocabulary size of 50,000 (approximately), and uses word embedding of dimension 1280 for learning the context for the words. The dataset is in English. GPT-2 large model has 762M model parameters and is sought to be one of the state-of-the-art techniques in NLP. The model has a total of 36 decoder blocks. GPT-2 model training on a large corpus of data and the model training is a self-supervised process i.e., with no human assistance or interference. The continuous input sequence of a certain length from the dataset is fed to the model and the same sequence with one token shifted to the right becomes the target to be predicted. The model predicts the next word for the given sequence of words to it. More precisely, for an input sequence, predictions for a token at 'i'th position only take the inputs from the token starting at position 1 to the token at position i and the subsequent tokens. As in the case of the traditional language models, the GPT-2 model outputs only one token at a time. After a token is predicted or learned, that token is appended to the sequence and again the sequence with the appended token is fed to the model and the process is repeated. The technique is termed autoregression. Therefore, GPT-2 models are also called "auto-regressive" models. The model uses the MLM (Mask Language Model) technique for learning the next word in sequence. The problem with the GPT2 text generation model is that the generated text is sometimes irrelevant, and the generated text doesn't make proper sense.

[7] The current approach in using pre-trained models for NLP tasks is to supervise the pre-trained model with the existing dataset for fine-tuning. Fine-tuning the model helps to understand the context for the downstream NLP task that we are going to solve. However, the GPT-2 model performance is impressive even without any further supervised training. The research concludes that the language model trained on large and diverse datasets like WebText becomes capable of performing well in different NLP tasks in different domains and datasets even without being fine-tuned.

[8] defined Data Augmentation as an approach to solving data-scarce situations by generating a synthesized training dataset using the existing training dataset to improve the model performance. [7] suggested that data augmentation for NLP is the process of generating synthesized utterances using the synonym word replacement technique which is more challenging as the generated utterances could be more invalid and fears that the augmented dataset could reduce the model performance.

[9] fine-tuned the pre-trained conditional BERT (Bidirectional Encoder Representations from Transformers) model for data augmentation using masked language modeling and used the intent category or class as a reference or condition for generating the utterance.

[7] suggests that there is a lack of enough research or theoretical underpinnings/principles backing up the fact that why DA works but the implementation of the DA concept with NLP tasks are being proven beneficial as shown by their different use cases. Another common issue that needs to be answered is the extent of augmentation to go for during data augmentation. Studies have shown that augmentation leads to a positive margin for the classifier. [10] suggests that the positive effect is seen in the model if the augmentation is applied exponentially. [11] suggests that unlike with Computer Vision, data augmentation in the case of NLP is challenging as the generated text is randomly distorted and semantically or grammatically incorrect or inconsistent. Data augmentation in NLP mainly follows word replacement using a synonym approach while augmentation of the utterance. However, the introduction of the GPT-2 model gave rise to a new approach i.e., using conditional and contextual data augmentation. They have also proposed a language model called LAMBADA (LAnguage Model BAsed Data Augmentation) which used Generative Pre-Training (GPT) model as the underlying algorithm and was able to generate augmented sentences based on the category (intent) name provided to that model. A clever approach followed by them was they obtained more suitable utterances out of the augmented dataset first by using the baseline mode i.e., only correctly classified sentences were selected from the augmented dataset. The proposed model was trained on the augmented training dataset and the model showed increased statistical performance when compared to the baseline model. The performance of LAMBDA was tested against different datasets and the resulting accuracy was compared with the accuracy coming from different models viz. CBERT, EDA, and the baseline model. In all cases, the LAMBDA outperformed with an appreciable difference. Fig. 1 shows the performance statistics for their experiment. [12] The paper studied the performance of different transformer-based pre-trained models such as GPT-2, BERT, and sequence-to-sequence model BART (Bidirectional Auto-Regressive Transformers) for conditional data augmentation with three different datasets. The research also studied the diversity of the dataset generated. The approach used seems to be alike to the work done by [11]. The target label for the utterance was appended to the utterance. The SEP (separation) token would separate the utterance and the label. The EOS (end of sentence) token would mark the end

Dataset		BERT	SVM	LSTM
ATIS	Baseline	53.3	35.6	29.0
	EDA	62.8	35.7	27.3
	CVAE	60.6	27.6	14.9
	CBERT	51.4	34.8	23.2
	LAMBADA	**75.7**	**56.5**	**33.7**
TREC	Baseline	60.3	42.7	17.7
	EDA	62.6	**44.8**	23.1
	CVAE	61.1	40.9	25.4
	CBERT	61.4	43.8	24.2
	LAMBADA	**64.3**	43.9	**25.8**
WVA	Baseline	67.2	60.2	26.0
	EDA	67.0	60.7	28.2
	CVAE	65.4	54.8	22.9
	CBERT	67.4	60.7	28.4
	LAMBADA	**68.6**	**62.9**	**32.0**

Figure 1 Experimental statistics for the LAMBDA model

Models	SST-2	SNIPS	TREC
No Aug	52.93(5.01)	79.38(3.20)	48.56(11.53)
EDA	53.82(4.44)	85.78(2.96)	52.57(10.49)
BackTrans	57.45(5.56)	86.45(2.40)	66.16(8.52)
CBERT	57.36(6.72)	85.79(3.46)	64.33(10.90)
$BERT_{expand}$	56.34(6.48)	86.11(2.70)	65.33(6.05)
$BERT_{prepend}$	56.11(6.33)	86.77(1.61)	64.74(9.61)
$GPT2_{context}$	55.40(6.71)	86.59(2.73)	54.29(10.12)
$BART_{word}$	**57.97(6.80)**	86.78(2.59)	63.73(9.48)
$BART_{span}$	57.68(7.06)	**87.24(1.39)**	**67.30(6.13)**

Figure 2 Model mean accuracy for different datasets

Models / Metrics	Genuine model	Combined (Genuine and Synthetic Model)
Precision	79.79	87.37
Accuracy	82.81	89.13

Figure 3 Yelp Pizza review classification report for models trained with original and synthetic datasets combined

of the complete utterance. 10 different variations were generated for each sentence present in the original dataset after the model was pre-trained with the existing dataset. Their research concluded that the sequence-to-sequence model i.e., BART outperformed as compared to the rest of the two pre-trained models. One issue seen during this experiment was preserving the label for the utterance seemed to be complicated. Fig. 2 shows the statistics obtained during their experiment.

[13] used the Yelp pizza review dataset which is an open research dataset along with GPT-2 transformer model and transfer learning approach to generate synthetic pizza reviews. The original dataset was smaller in size and was a balanced dataset. The dataset consisted of reviews on pizzas in the English language and the reviews were categorized as positive or negative reviews. The original dataset consisted of a total of 450 observations but was increased to 11380 after applying augmentation. First, the model was fine-tuned with the Yelp pizza dataset. Then the fine-tuned model was used to generate a synthetic pizza review dataset. The original dataset was combined with the augmented dataset to construct the final dataset. Certain data was obtained from the original dataset to benchmark the performance of the baseline model and the newly trained model on the augmented dataset. The baseline model's overall accuracy was found to increase from 82% to 89% in the new model trained with the augmented dataset as shown in Fig. 3. They suggested fine-tuning the model before implementing the dataset augmentation approach during their experiment.

[14] has introduced a multitask transformer model called DIET (Dual Intent and Entity Transformer) that achieves the two prime requirements in language understanding (NLU) in NLP: intent detection and entity extraction jointly. The research has brought the fact into light that modeling those two sub-tasks using different models can suffer from error propagation but implementing those two tasks using a multi-task architecture is beneficial in terms of achieving task efficiency. The Rasa team also recommends using the DIET classifier as it outperforms the other intent classifier algorithms. The architecture of Rasa is highly customizable where many components can be selected or adjusted in a plug-and-play fashion, components can be activated or deactivated. There are various features of DIET architecture. Rasa itself does not provide any pre-trained weights but allows the user to specify dense features with the use of pre-trained language tokenizers and featurizers. Secondly, the sparse word or n-gram featurizer can also be added to the model pipeline in a plug-and-play fashion. The research has also demonstrated that the DIET models can perform much better even with the specific domain-related dataset. Using DIET models in purely supervised settings can outperform fine-tuned BERT models with a significant reduction in model training time.

IV. Solution Architecture

A. *Architecture for SkillBot chatbot application*

Most of the chatbot frameworks are found to support the

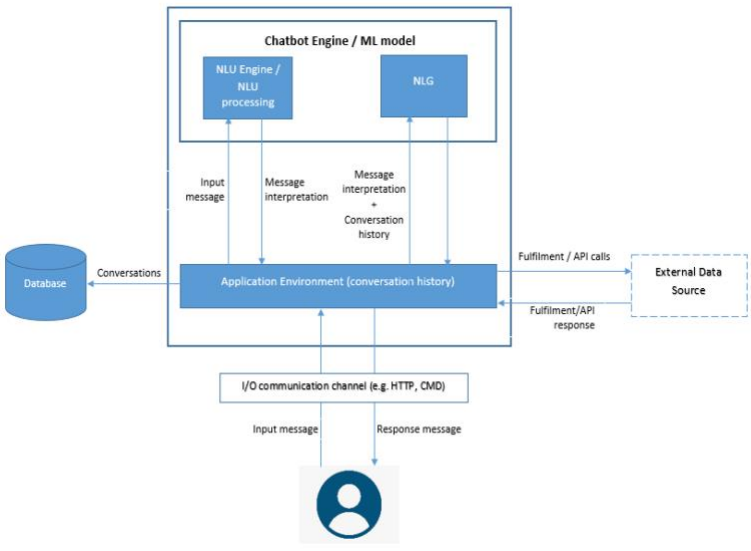

Figure 3 Architecture for chatbot application

architecture shown in fig. 4 for their chatbots. Users communicate with the chatbot over the I/O channels such as HTTP REST API. The chatbot software mainly has two ML components: NLU and NLG. When the chatbot receives a message from the user, it is sent to the NLU section for message parsing or message interpretation. Intents, entities etc. are interpreted and sent back to the host environment. Then the interpreted results and conversation history are sent to the NLG section. The NLG then refers to the history object and patterns it learned during the training phase and determines which action to take first. If actions are text responses from NLG, then NLG forwards the response to the host environment. However, if the action determined is a fulfillment or API call then the host environment performs these actions and forwards the received response back to the user over the communication channel. In our case, we are not using any fulfillment or external API calls as a response. Both components of NLU and NLG must be trainable as well. Additionally, chatbot software can also be configured to store all the conversations, every single detail associated with the conversation, in a database. In our case, we configured a database for storing all the conversations. Fig. 4 represents the architecture for basic ML-based intent detection-based chatbots application as suggested in the paper [15].

[16] Similar to other chatbot systems, there are two main components in the Rasa framework: Rasa NLU for Natural Language Understanding (NLU) and Rasa Core for Natural Language Generation (NLG). Those components are decoupled (independent of one another) but still work hand in hand in the Rasa framework. Rasa NLU is used for natural language understanding i.e., perceiving or predicting the intent based on the user-sent message and Rasa Core is a dialogue management system that is responsible for predicting suitable response or action for the predicted intent by Rasa NLU. Those components are defined in plug-and-play fashion in the configuration pipeline since the

architecture in Rasa is modular by nature and exposing HTTP APIs is possible with these components. Also, since Rasa NLU and Rasa Core are decoupled, they can be implemented independently of one another. In the case of the Rasa chatbot all tools and technologies we use to construct the NLU, and NLG model can be specified using the configuration inside the file called *config.yml* file. Fig. 5 shows the configuration we applied for our chatbot application. The components used for the NLU section are listed under the section called *pipeline* and the components used for the NLG section are listed under the *policies* section. We used the DIET classifier for NLU and TED policy for the NLG section.

B. DIET as an NLU algorithm

Rasa allows pre-trained weights to be used but in our case, we are not using any dense features. The architecture will refer to the sparse embeddings from the *CountVecot featurizer* in the pipeline. Intent classifier and entity recognition. We are using both of these components, therefore we set the keys *intent_classification* and *entity recognition* to be true.

```
language: en
pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: DIETClassifier ···
- name: EntitySynonymMapper
- name: ResponseSelector
  epochs: 120
- name: FallbackClassifier
  threshold: 0.4
  ambiguity_threshold: 0.1
policies:
- name: MemoizationPolicy
- name: TEDPolicy
  max_history: 5
  epochs: 120
- name: RulePolicy
```

Figure 4 configuration used for the chatbot application

The model is not using any pre-trained weights or dense features. There are two layers of transformers. The encoders and decoders are identical to each other but maintain different weights. In addition, the decoder block additionally contains a multiple. The encoder models have an attention layer which is implemented as a multi-head attention model consisting of 4 attention heads for multiple input or context representation inspection for input tokens. The input embedding size received is 256 dimensions for the encoder. The learning rate is set to 0.001. The training epoch set is 120. The loss calculation will be done using the SoftMax function at the output layer. Fig. 6 shows the hyper-parameter settings used while implementing the DIET model for our *SkillBot* chatbot. The settings have been defined inside the *config.yml* file inside the root application folder for the chatbot application. During training, at first, input sequences are tokenized and a dictionary using words and sub-word (n-grams) is built. The *WhiteSpace* tokenizer and *CountVector featurizer* specified in the pipeline do this job at the beginning. The model will be initialized with the sparse feature matrix before the training of the components. [17] The input entering the DIET model consists of tokens as shown in fig. 7.

```
- name: DIETClassifier
  featurizers: []
  share_hidden_layers: false
  loss_type: softmax
  ranking_length: 10
  transformer_size: 256
  number_of_transformer_layers: 2
  number_of_attention_heads: 4
  epochs: 120
  learning_rate: 0.001
  use_masked_language_model: false
  use_key_relative_attention: false
  use_value_relative_attention: false
  intent_classification: true
  entity_recognition: true
  batch_size:
  - 64
  - 256
  batch_strategy: balanced
  embedding_dimension: 20
  dense_dimension:
    text: 128
    label: 20
  concat_dimension:
    text: 128
    label: 20
  regularization_constant: 0.002
  negative_margin_scale: 0.8
  drop_rate: 0.2
  drop_rate_attention: 0
  weight_sparsity: 0.8
```

Figure 5 Format for input embeddings entering DIET model

Utterances tokens	Entities token (if exists)	Inent label	CLS token	MASK token	Masked token

Figure 6 Format for input embeddings entering DIET model

```
- name: TEDPolicy
  max_history: 5
  epochs: 120
```

Figure 7 TED policy configuration

C. TED policy as an NLG algorithm

[18] TED is a transformer-based architecture for dialogue policy. TED stands for Transformer Embedding Dialogue policy. The self-attention mechanism used by the transformer model has outperformed the task of next action prediction as compared to traditional hierarchical RNN networkS. Contrary to RNN (Recurrent Neural Network) which equally prioritizes the elements in a sequence, the TED algorithm with the use of the attention mechanism can select the most appropriate next action at a time 't' by referring to the stack of feature tokens embedding produced at time t-1, t-2… and t-n according to the number of hyper-parameters set for retaining the dialogue stack. If the model at time 't' receives an irrelevant input, then the model can ignore the input and determine the next suitable action based on the embeddings present in the stack in the previous timestamp. The sequence of token embeddings present in the stack or the history object could represent multiple topics or intents but the transformer model with its self-attention mechanism can learn to resolve or satisfy those intents with an appropriate response in due course of time. The TED policy for our project has been used in a modular fashion. Fig. 8 shows the configuration we used for implementing the TED policy for our project. TED model is trained jointly with features coming from NLU (i.e., token in the form of a concatenated array of intent category, entity label, and token embedding for the previous action), and sequence of tokens present in the stack and the token representation of the action as specified in the story. Later, during the time for inferring a particular action, the predicted action token is compared with every token of actual actions. Then a highly similar action is determined as the next action to be executed.

As suggested by [18], fig. 10 shows that the Input token at time 't' on the left side of the block diagram is the concatenated input token predicted from the NLU component. Similarly, on the right side, the actual token embedding for the action as specified in the story is passed as the target. We have set the transformer size to 5 which is a hyper-parameter. Therefore, there are 5 different layers in the diagram which serve to implement a stack mechanism for storing the past embeddings. As suggested by [18], fig. 9 shows the constituents for the input token entering the TED transformer block where key, value, and query vectors are present and are randomly initialized. The token embedding size coming from the transformer is larger. The tokens coming from the transformer are fed to the feed-forward dense network which learns some weight and the feedforward network outputs the reduced token size. At the same time, the target token embeddings are processed by the feed-forward network. The newly learned embeddings for the actual token and the predicted embedding tokens reach the similarity check layer where the similarity difference is calculated in the form of dot product loss. This loss is propagated back to feed-forward layers and the transformers in the form of error gradients and those feed-forward layers and transformer layers learn the appropriate weights. During runtime, the input target section on the right receives the list

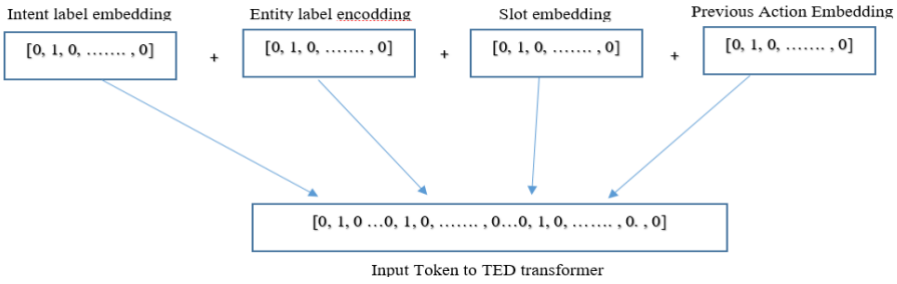

Figure 9 Input token at time 't' entering the transformer network inside the TED model

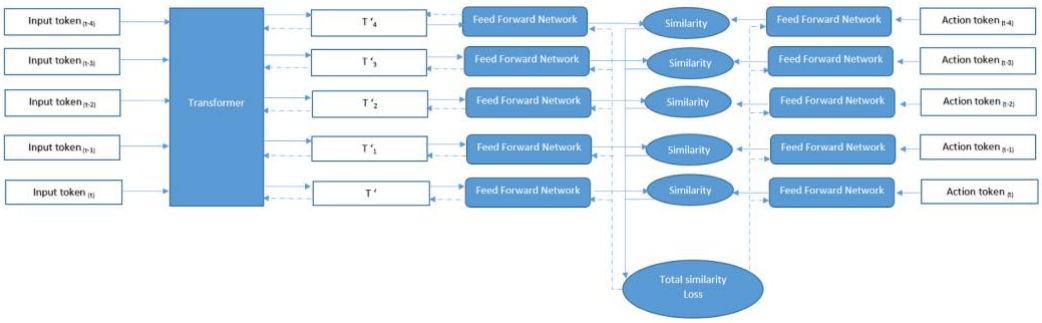

Figure 8 TED model during training

1. Load and Initialize **GPT-2 model**
2. Load and initialize **pre-trained tokenizer (data encoder)**
3. Read the original dataset as **D**
4. Create empty augmented dataset object as **D_aug**
5. REPEAT for each Intent, **I**: **I** ϵ **D**
    a. Read list of utterances into object **U**
    b. List of input embeddings **T** = **tokenizer (U)**
    c. Augmented utterances list **U_aug** = **GPT2(T**, sample_num=**15**, max_words=**20**)
    d. Set KEY= I and VALUE= append (**U**, **U_aug**)
    e. Append {KEY, VALUE} to **D_aug**
6. Save **D_aug** to a file

Figure 10 Pseudocode for data augmentation

of token embeddings for all the actions. The predicted similarity for input tokens is checked against the list and the most similar action token is selected and the respective action is executed. Therefore, the TED policy is still a retrieval model rather than a new content generation. The learning on sequenced-based action embedding learning provides the Rasa NLG with better generalization capability for the actions as compared to other approaches using RNN and LSTM (Long Short-Term Memory cell) networks.

D. Pseudocode for data augmentation

The task of augmenting the dataset can be automated using a program script using programming languages that support the library for using transformer models. In our case, we used Python programing language for scripting the task of data augmentation for our dataset.

Fig. 11 shows the pseudocode for augmenting the dataset. First, we initialize and load the GPT-2 model and pre-trained tokenizer into the memory from respective libraries. To augment the dataset, first, the existing dataset files are read. In our case, we used Python language, and we used the *transformer* library from which we created a *pipeline* object which is an abstraction, and encapsulates the task of loading and initializing the model and the tokenizer supplied as named arguments. Rasa training dataset files have YML format. Those YML data need to be read into native programming data-structure objects supported by the programing language which is used for writing the script. In our case, we read the dataset into a dictionary object as *D*. We then create an empty dictionary object for holding the augmented dataset and named it *D_aug*. There are multiple intent categories with multiple utterances in a single file. We loop through each intent *I* and read the list of utterances present under the intent *I* into the list called *U*. We then provide the list *U* to the tokenizer for creating their corresponding input token embeddings and read the encoded embeddings into another list object called *T*. We obtain the numeric representations as embedding from the tokenizer for each utterance. GPT2 model upon receiving such input embedding generates (predicts) contextualized utterance which is a conditional open-end text generation approach. The list of tokens is supplied to the *GPT2* model which receives each group of each token embedding as an input and returns 15 different variants with each variant consisting maximum of 20 words. The augmented utterances are read into a list object called *U_aug*. The augmented *U_aug* is appended to *U* i.e., original utterances and augmented utterances are combined. Then the intent *I* and the combined utterances are stored as key-value pair objects. The key-value pair created is appended to the empty augmented dataset object we created before i.e., *D_aug*. The same process is repeated for all intent categories present in the training dataset file. Finally, we must convert and write the *D_aug* dataset object into YML format.

Before applying dataset augmentation, we manually removed the inconsistencies by removing similar utterances from multiple intents to make our dataset as accurate as possible which was time-consuming. We used a Google Colab environment with GPUs for the implementation of the task for dataset augmentation since the process of encoding and decoding the text embeddings and generating a bigger number of variations took a much longer time with the use of CPUs only. Therefore, the use of GPU expedited the task, and the task of DA became quicker.

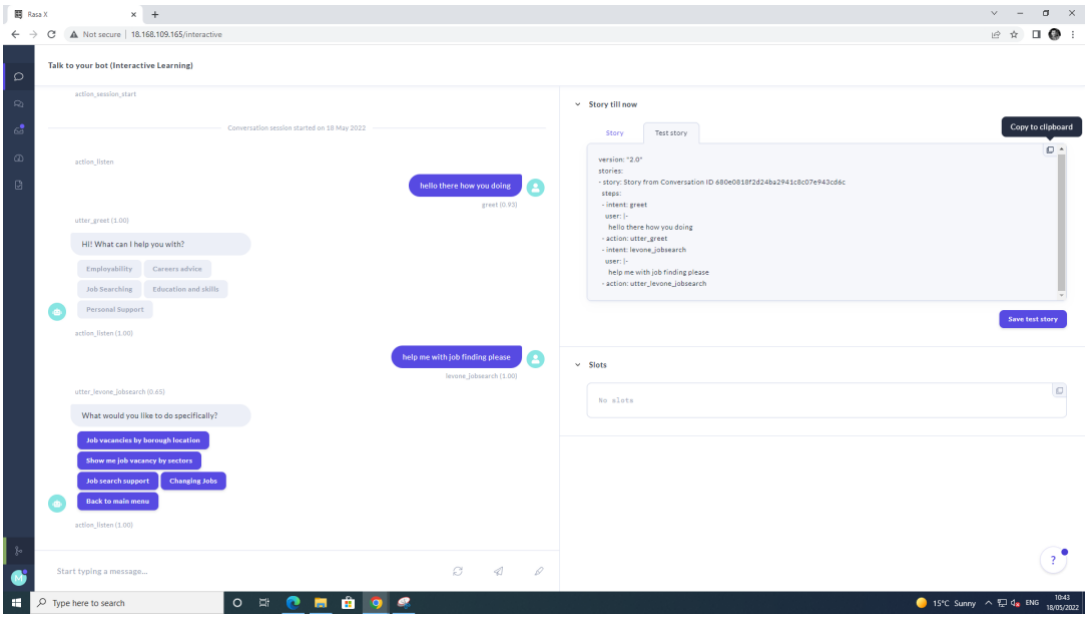

Figure 12 locating the test story for a conversation on Rasa X app

V. Performance Analysis and Evaluation.

Testing a language model is a difficult task because of the non-discrete nature of the language data. [19] The chatbot application can be assessed using five different evaluation metrics related to user experience, information retrieval process, linguistic capability, business, and technology. We are assessing information retrieval capability in our case by assessing the dialog management system of the chatbot. We acquired two versions of the datasets: the original datasets and the augmented datasets. We trained the Rasa chatbot model separately using both datasets and obtained two different NLP models. We treated the chatbot model trained on the original dataset as the baseline model and the chatbot model trained on the augmented dataset as the proposed model. The pipeline configuration for both of those models was kept the same; only the dataset was different. Rasa provides a specific set of CLI (Command Line Interface) tools to facilitate the chatbot test. Rasa has two test approaches available: NLU tests and story-based tests. NLU test only tests the performance of the intent classifier and entity extractor of the chatbot but the story-based test approach tests both: NLU and NLG components of Rasa simultaneously. Hence using the story-based approach we can measure the performance of both components at the same time. We have used a story-based test approach for chatbot model evaluation as this would test both NLU and NLG sections in an end-to-end fashion as discussed here[20]. An attempt was made to include a story covering most of the intents present in both versions of the dataset for evaluation purposes, which was the recommendation from the Rasa team for testing. For example, intents representing the Smalltalk conversation were not included in test stories because the Smalltalk was not present in the earlier state of the dataset. We evaluated the performance of those individual chatbot models against the common 143 test stories to benchmark their dialog management system. Then we also compared the two performances.

A. Generation of test stories

Test stories (test cases in our case) were prepared manually by reviewing the existing conversation and picking up only the conversation with one level of turn from Rasa X. Fig. 12 shows an example of obtaining a Test story using the Rasa X GUI (Graphical User Interface). The left section is the actual conversation taking place between the user and the chatbot and the right section shows the sequence of intent and utterance generated during the conversation. The Test Story tab was clicked, and the test story was copied. The story was saved inside the *test_stories.yml* file.

```
version: "2.0"
stories:
- story: Story from Conversation ID 1
  steps:
  - intent: levone_caradv
    user: |-
      is computing good career choice
  - action: utter_levone_caradv

- story: Story from Conversation ID 2
  steps:
  - intent: levthree_carkeysec_comp
    user: |-
      I mean is career in computing a wise decision
  - action: utter_levthree_carkeysec_comp


- story: Story from Conversation ID 3
  steps:
  - intent: gratitude
    user: |-
      thanks
  - action: utter_gratitude

- story: Story from Conversation ID 4
  steps:
  - intent: levfour_carkeysec_railway
    user: |-
      I want to work at London underground
  - action: utter_levfour_carkeysec_railway
```

Figure 13 Content inside test_stories.yml file

Fig. 13 shows a snippet of how the *test_stories.yml* file looks in terms of the content it holds. A total of 143 stories each consisting of one intent and one utterance were picked. The file was placed inside the *tests* folder inside the main application directory for the Rasa chatbot. We separately tested the same test stories against the two models using the command: *rasa test*. The command evaluates the dialog management system and reports performance metrics for both Rasa NLU and Rasa Core components.

Table (1) shows the overall score for the selected metrics. The problem we tried to solve is text classification with multiple class or intent categories and the test has summarized precision and f1 score. Both models showed overall precision of 100% but since this is a multiclass problem, we should focus on the F1 Score i.e., the higher the F1 score, the better the model is. The improved F1 score suggests that the proposed model is a good model with improved generalization capability. The baseline model executed only 38.5 % of the stories successfully whereas the new model successfully executed 70.6% of stories i.e., successful run of the story from the start till the end. The F1 score for NLU also improved from 55.6% to 82.8%.

Table (2) shows the overall precision and F1 Score for NLG for the baseline model and the proposed model. The metrics for the proposed model have been improved significantly as compared to the baseline model for the NLG

Table 1 Comparison of NLU performance

Chatbot model	Precision (Average)	F1-Score (Average)	Story Accuracy
Baseline model	100%	55.6%	38.5%
Proposed model	100%	82.8%	70.6%

Table 2 Comparison of NLG performance

Chatbot model	Precision (Average)	F1-Score (Average)	Generated Action Accuracy
Baseline model	85.7%	79.2%	77.9%
Proposed model	93.2%	90..3%	90.2%

component. This is obvious because, in the case of the proposed model, NLU performance has been increased. Improved NLU performance promotes improved NLG performance as the NLG component uses the output from the NLU component in predicting the next output and if the NLU is more accurate, the more the accuracy of NLG becomes.

The intent classification report or confusion matrix for the NLU section is too large to fit into the document because of the existence of a huge number of categories and hence not included here.

VI. Conclusion

It is learned that the use of transformer-based models in the field has expanded the horizon of the NLP domain. One of the areas that are under research is the use of transformer-based models in the field of data augmentation in NLP. The task of data augmentation is challenging and may require a great amount of time and effort. More training data means more training steps and more training steps means more model training time. The augmented dataset demanded more training time for the model. GP2 model seemed to be efficient in the text generation process if the prompt supplied is imperative the tone of the response generated would be more contextual or more sensible but if the prompt supplied is affirmative such as a title for a paragraph the nature of the response generated text in such a case makes less sense or irrelevant. Using an ML model for data augmentation can be helpful and cast a positive impact but dataset augmented using machine learning models still must go through some review processes before they can be incorporated into an application as an integral part of the dataset. It has been concluded that the chatbot can be used to facilitate different tasks for humans and simplify interaction with the computer, but they are not aimed to replace humans. During the development project, a different set of tools and technologies and their technical implementation were learned. The development of the dissertation development process became knowledge-gaining and insightful.

VII. Future Work

We did not fine-tune the GPT2 model before we used it for purpose of data augmentation. We used open-end conditional text generation which left the possibility of generating irrelevant texts. Therefore, considering every single augmented utterance without reviewing them manually or in some automated fashion would cause the addition of noise to the model training process which could have contributed to model overfitting which in turn can impact the overall model accuracy. Therefore, we could apply and experiment with a few more approaches. First, we could fine-tune the GPT2 model using our original dataset which would improve the quality of the text generated i.e., could make them more contextual. Further, we can reduce the effort of reviewing the augmented dataset by filtering (classifying) the augmented dataset with the use of another fine-tuned transformer model e.g., BERT. We could fine-tune BERT with the original dataset along with the intent categories and then use the same BERT model for classifying the augmented dataset. Then we can only take the correctly classified dataset to review. Using the filtered dataset obtained this way could further increase model accuracy. We could also experiment with the bi-directional text generation model such as BART for creating a different version of the dataset and reassessing those performance metrics for both NLU and NLG components of SkillBot chatbot.

Acknowledgment

The work presented in this paper is carried out as a part of SkillBot – a chatbot-based job market intelligence tool. The project has been funded by the Mayor and Burgesses of the London Borough of Lambeth.

References

[1] A. Radford et al, "Language models are unsupervised multitask learners," OpenAI Blog, vol. 1, (8), pp. 9, 2019

[2] NLU Training Data. Available: https://rasa.com/docs/rasa/nlu-training-data/.

[3] K. White, "10 Best Practices for Designing NLU Training Data," 2020

[4] R. Prabhakaran, "Smalltalk for LATEST Rasa Stack," 2020.

[5] A. Vaswani et al, "Attention is all you need," Advances in Neural Information Processing Systems, vol. 30, 2017.

[6] R. Adolphs, "Social cognition and the human brain," Trends Cogn. Sci. (Regul. Ed.), vol. 3, (12), pp. 469-479, 1999.

[7] S. Y. Feng et al, "A survey of data augmentation approaches for nlp," arXiv Preprint arXiv:2105.03075, 2021

[8] S. C. Wong et al, "Understanding data augmentation for classification: When to warp?" in 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), 2016.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018

[10] S. Rajput, Z. Feng, Z. Charles, P.-L. Loh, and D. Papailiopoulos, "Does data augmentation lead to positive margin?," 2019, pp. 5321–5330

[11] A. Anaby-Tavor et al., "Do not have enough data? Deep learning to the rescue!" 2020, vol. 34, no. 05, pp. 7383–7390.

[12] V. Kumar, A. Choudhary, and E. Cho, "Data augmentation using pre-trained transformer models," 2020.

[13] D. Whitfield, "Using gpt-2 to create synthetic data to improve the prediction performance of nlp machine learning classification models," 2021.

[14] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol, "Diet: Lightweight language understanding for dialogue systems," 2020.

[15] E. Adamopoulou and L. Moussiades, "An overview of chatbot technology," 2020, pp. 373–383.

[16] T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol, "Rasa: Open-source language understanding and dialogue management," 2017.

[17] Rasa, "Components," 2022. https://rasa.com/docs/rasa/components/.

[18] V. Vlasov, J. E. Mosig, and A. Nichol, "Dialogue transformers," 2019.

[19] G. A. Santos et al, "A Conversation-Driven Approach for Chatbot Management," IEEE Access, vol. 10, pp. 8474-8486, 2022

[20] T. Bocklisch et al, "Rasa: Open source language understanding and dialogue management," arXiv Preprint arXiv:1712.05181, 2017.