LET'S AGREE TO DISAGREE, MANY2ONE ENSEMBLE MODEL FOR REALISTIC ONLINE CHAT BOTS

Agamdeep S. Chopra
Graduate Research Assistant
Department of Computer Science, Stevens Institute of Technology
Hoboken , NJ
achopr4@stevens.edu

December 15, 2021

ABSTRACT

Through this course project my goal is to learn about ensemble techniques for NLG problems and try to suggest possible novel improvements, using various NLP models learnt in class.

1 INTRODUCTION.

The goal of this project is to implement and analyze various NLP models and ensemble techniques to improve chat-bot text generation and result in a more realistic human-like chat experience. This project was motivated by my first-hand experience trying to develop a friendly Customer Service chat bot for my former workplace. Training and fine-tuning our model was extremely tricky, and after numerous failures, we decided to implement a simpler model using lookup tables for efficiency.
To produce a human-like chat experience a model must produce relevant text, that is coherent, should not produce generic responses, should not repeat previously resolved dialogues in a chat session, and have a consistent persona throughout a chat session. Recent publications, which will be discussed later, have tried to resolve these issues but due to time limitations and hardware restrictions, such implementations are out of the scope of this project. Instead, we will develop 3 simple seq-2-seq models using our data set, implement and evaluate 1) a traditional ensemble technique and 2) try to outperform this technique using an automatic-ensemble model, and finally evaluate results from this experiment and propose possible solutions.
[Note that the goal of this project is not to achieve state-of-the-art performance but to simply test if our hypothesis has a promising indication to address any of the stated problems with chat-bots, and if not, try to theorize any possible solutions.]

2 BACKGROUND.

Currently, producing realistic human-like language generation is an open problem. Recent publications suggest encoding dialog history and external knowledge as in Heterogeneous Memory Networks [Zehao Lin et al., 2019] produce realistic responses to preserve context and dialog continuity. The authors of [Thomas Wolf et al., 2019], suggest a novel model architecture instead of traditional seq-2-seq to address personality inconsistency, lack of long term memory, and generic responses. While the authors of [Zhaojiang Lin, et al., 2019] suggest a model that can recognize emotions and produce appropriate responses. But, all of these models suffer

from additional model complexity and computational overhead. The authors of [A. Aniol et al., 2019, pp. 180-183] suggest using a simple ensemble based approach that assignes weights to each model using F1 scores calculated over a pre-evaluation set, to deliver a better language generation performance. This is done by picking the response that has the highest cumulative weight. The equation of cumulative weight is given by the following equation:

$$w_m = \sum_{i=1}^{M} w_i$$

Eq. 2.1. Cumulative weight.

Where w_m is the cumulative weight of m-th unique response dialog, w_i is weight of a model whose predicted response equals the m-th unique response, and M is the total number of models with the same m-th unique output. The downfall of this method is that for a small number of models, if the F1 scores do not add up to be greater than that of the highest F1 scored model, the ensemble algorithm will pick the output of the highest F1 model every time regardless of it being correct or being the majority prediction. For the scope of this project, I evaluated this approach and tried to suggest any improvements. Even though this approach does not address any of the complex problems discussed earlier and there do exist models as discussed earlier that reach state-of-the-art performance in human-like dialog generation, due to hardware and time restrictions, this method was chosen.

3 APPROACH.

It was the goal of this project to evaluate the ensemble method discussed in section 2 and try to improve this it by designing an automatic ensemble model to give better performance on the evaluation metrics, F1 score and bleu score, for chat-bot dialog generation. All the models were designed and optimized from scratch. Initially, my goal was to fine-tune pre-trained state-of-the-art NLG models. But, due to hardware limitations, I switched to training my simple seq-2-seq models. After training these models, global F1 scores were calculated for each model and used to implement the ensemble methods. Lastly, a simple auto-ensemble model was developed, but due to time limitation, this model was kept as simple as possible. Finally, using the results of the experiment and the recent publications mentioned in section 2, a preliminary black-box model is suggested(discussed in section 6).

4 EXPERIMENTAL DESIGN.

This project was coded in Python3 from scratch, primarily using Cuda-enabled PyTorch libraries. Due to the complex nature of this problem, cross-validation failed to give useful hyperparameters and hence was discarded. Instead, the models were manually optimized after every 20, 10, 2, 2, 2, 2 epochs successively. Preferably, I would like to leave such tuning to be automatic but due to limited time, this was done manually. All base seq-2seq models, i.e. LSTM, CNN, and Transformer(ATT), were evaluated on the pre-validation set using a custom F1 score $[0, 1]$ algorithm(this algorithm calculates the global F1 scores by calculating Precision and Recall), that were used as weights for the ensemble technique discussed earlier. The ensemble technique and auto ensemble model was evaluated on the validation set using nltk's sentence bleu score $[0, 1]$ function. Most models took about half a day to train and fine-tune. This can be lowered with larger batch sizes if enough GPU memory is available. Unfortunately, due to hardware restrictions, I trained most models with batch-size of 32 on custom train functions for each model. The following sections give a deeper insight into the data set, models, and other miscellaneous settings.

4.1 Data set

I utilized the Daily Dialogue data set [Li et al, 2017] provided by the Huggingface Dataset library. This was used to extract approximately 75,000 dialog pairs. To keep the dialog generation as natural(referring

to the assumption that real human conversations have many spelling and grammatical errors) as possible, no text cleaning was performed during prepossessing. This raw set was used to generate a vocabulary that also contained <sos>, <eos>, <unk>, and <ignore> tags and custom tokenizer, token to word, id to one-hot, and one-hot to id functions were defined to process the data. A resize function was defined that padded or cropped each dialog to the mean word length of the data set. Small dialogues were padded with the <ignore> tag, and large dialogues were cropped from the beginning assuming that the most important context for a response generation task lies in the final few words of a long dialog. This was done by manually examining the longest dialog sequences and their expected response in the data set. The purpose of the <ignore> tag is for the attention-based mechanisms used in some of the models. The indices of <ignore> tags in the ground truth response are not used for loss evaluation or metric evaluation. The <sos> tag signals the start of a sentence and is the first tag passed to the decoder by default. The <eos> tag signals the end of a sentence for both encoder input and decoder text generation. The <unk> tag is assigned to any unseen word not indexed in the vocabulary. The data set is divided into 3 categories: train set(used to train the models), pre-validation set(used to evaluate the F1 scores of the base models), and validation set(used to evaluate the bleu scores of the ensemble methods).

4.2 Base Models

I trained 3 base seq-2-seq models as a building block for this project. All 3 models were trained using cross-entropy loss(ignore index = index of <ignore>, reduction = mean) as the criterion and ADAM optimization. All models were trained on a Cuda-enabled GPU.

4.2.1 LSTM

The LSTM model consists of an encoder and decoder, each with its own embedding and n LSTM layers. During initialization, dropout was set to 0.1, embedding dimension was set to 256, hidden dimension was set to 512, and the number of layers, n was set to 5 for both the encoder and decoder. The batch size was set to 32, L2 regularization was set to $1E-9$, amsgrad was set to True, teacher forcing ratio was set to 0.5 initially(lowered by 0.1 each learning rate step down), 20 epochs initially(followed by 10, 2, 2, 2 corresponding to learning rate step down), and the initial learning rate was set to $1E-4$ stepping down based on the loss values. For my final run the learning rates were in the order $1E-4$, $1E-5$,$1E-6$,$1E-9$, and $1E-10$. Perhaps with more time and better hardware, the LSTM model could be made more complex and fine-tuned more precisely. During validation and evaluation, the model is set to evaluation mode and all gradients are turned off. With the teacher forcing ratio set to 0, the model was used to generate the predicted responses.

4.2.2 CNN

The CNN model follows the Convolutional Sequence to Sequence Learning [J. Gehring et al., 2017]. This model utilizes a fully convolutional architecture for the encoder and decoder with attention modules. During initialization, dropout was set to 0.1, input dimension was set to vocab length, output dimension was set vocab length, embedding dimension was set to 256, hidden dimension was set to 512, kernel size was set to 3, padding was set to the index of <ignore> tag, and the number of layers, n was set to 1 for both the encoder and decoder. The batch size was set to 32, L2 regularization was set to $1E-6$, amsgrad was set to False. The model was trained for 20 epochs initially(followed by 10, 2, 2, 2 corresponding to learning rate step down), and the initial learning rate was set to $1E-2$ stepping down based on the loss values. For my final run the learning rates were in the order $1E-2$, $1E-3$,$1E-4$,$1E-5$, and $1E-6$. The model converged relatively fast after some hyperparameter tuning and gave decent predictions. During validation and evaluation, the model is set to evaluation mode and all gradients are turned off.

4.2.3 Transformer

This attention-based model follows the architecture of a Transformer [A. Vaswani et al., 2017]. During initialization, dropout was set to 0.1, input dimension was set to vocab length, output dimension was set vocab

length, position-wise dimension was set to 512, embedding dimension was set to 256, hidden dimension was set to 512, the number of heads was set to 8, and the number of layers, n was set to 3 for both the encoder and decoder. The batch size was set to 32, L2 regularization was set to $1E-6$, amsgrad was set to True. The model was trained for 10 epochs initially (followed by 4 and 10 epochs corresponding to learning rate step down), and the initial learning rate was set to $1E-3$ stepping down based on the loss values. For my final run, the learning rates were in the order $1E-3$, $1E-5$, and $1E-9$. The model converged extremely fast with excellent predictions. During validation and evaluation, the model is set to evaluation mode and all gradients are turned off.

4.3 Ensemble

The ensemble method uses the calculated F1 scores of the 3 models to pick unique responses with the highest cumulative weight as discussed in section 2. This is done using various conditional statements. The final outputs of this method are evaluated using the bleu metric.

4.4 Many2One Ensemble / Auto-Ensemble

In this technique, I developed a simple CNN model with 2 1D convolutional layers, that iterates over stacked predictions of each of the models such that at any instant of the iteration, 3 words, one from each predicted output, from the same index location are passed to the model to generate an ensemble prediction. Unfortunately, due to time constraints, I was unable to fine-tune or properly optimize the parameters of this algorithm. During training, the model's learning rate was set to $1E-3$ for 15 epochs, with L2 regularization set to $1E-9$, amsgrad set to True, and batch size set to 64.

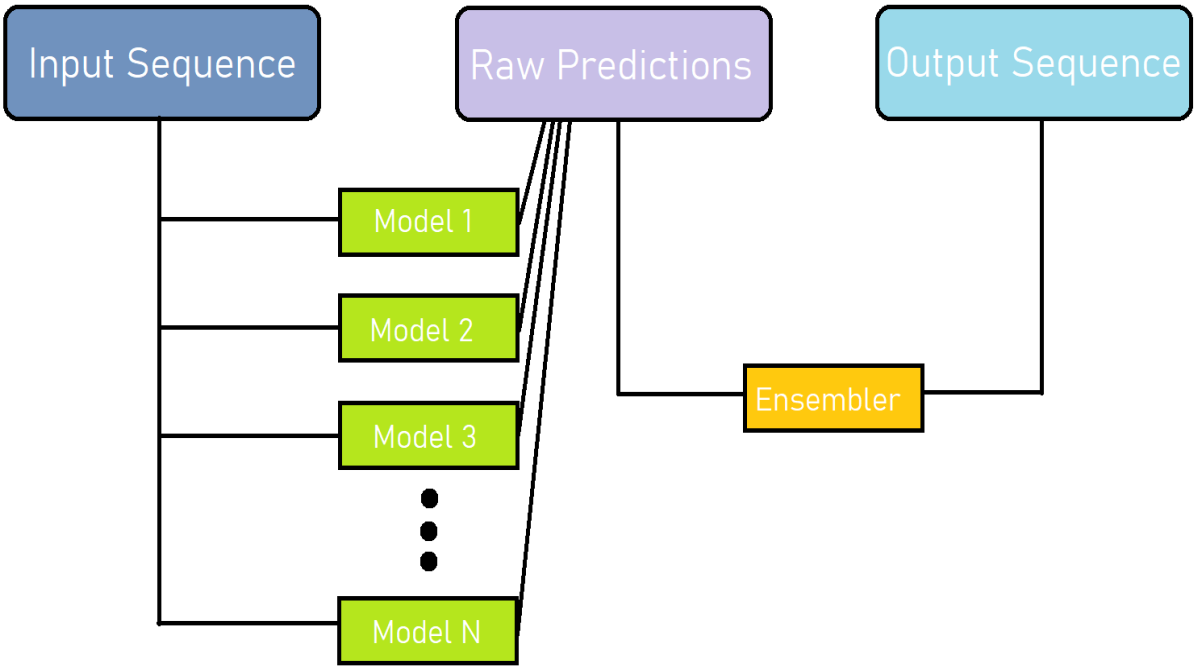

Fig.4.4.1. Black-box auto-ensemble model.

4.5 Validation Metrics

F1 metrics were calculated for the base models by first counting the total true positives(TP), false negatives(FN), and false positives(FP) in the data set for each sentence's index except the <ignore> tag indices for the ground

truth. This is done by trimming each dialog until the <eos> tag is reached in the ground truth. Using this information, Precision(P) and Recall(R) are calculated using the formula:

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

Eq. 4.5.1. Formula for Precision and Recall.

The precision and recall values were then used to calculate the overall F1 score of each model using the equation:

$$F1 = \frac{2 \cdot P \cdot R}{P + R + \epsilon}$$

Eq. 4.5.2. Formula for F1 score.

where $\epsilon = 1E - 12$ to prevent dividing by 0. The bleu scores were evaluated for the ensemble models using nltk package's sentence bleu function per dialog response. The score was then averaged to give an average bleu score for the model. These metrics help us quantify how well the models are performing on the validation data set. While F1 measures uni-gram statistics, the bleu score measures the uni-gram, bi-gram, tri-gram, n-gram statistics. Thus while F1 score is a good indicator for accuracy on a word level basis, bleu lets us measure how well the generated text matches a real human-generated dialog.

5 EXPERIMENTAL RESULTS.

In the table below, the transformer model outperformed all other base models. Its F1 score was greater than that of the CNN and LSTM model combined, thus causing the ensemble algorithm to only pick its outputs as described in section 2. Thus, for a small number of base models, if the F1 scores are not comparable, the ensemble algorithm fails and produces generic Transformer outputs during dialog sessions suffering from continuity and persona issues. On the other hand, the Transformer/Ensemble model achieved the highest bleu score and surprisingly the auto ensemble model performed significantly worse. This may be due to the fact that I was unable to properly optimize the model due to time constraints or a more powerful model is needed. Perhaps using an attention-based model might boost performance and lead to faster convergence for the auto ensemble technique. The outputs of this technique were semantically incoherent and broken.

Observed metrics		
Model	F1 score	BLEU score
LSTM	0.018	-
CNN	0.857	-
Transformer	1.0	0.952
Ensemble	-	0.952
Auto-Ens.	-	0.283

Table. 5.1. F1 score and BLEU score results for the experiment.

Neither of the models or techniques implemented was able to address the issue of non-human-like chat-bot behavior. Since these models were fairly simple, using a small data set and vocabulary, they lacked any internal mechanism or training technique to produce more interesting and vibrant realistic human-like dialogues. All models took a very long time to optimize with the exception of the Transformer that converged relatively quickly, showing the power and efficiency of pure attention-based algorithms. The CNN model performed decently but the LSTM model failed to produce any meaningful dialogues. Perhaps, by replacing the LSTM model with a stronger model, better performance can be achieved for the ensemble techniques. The auto-ensemble technique, although promising, only uses uni-gram-based output generation which may explain its failure to generate meaningful and semantically coherent dialogues. The auto-ensemble model also introduces an additional layer of unnecessary complexity, which in hindsight is not worth the added computational cost.

6 CONCLUSION AND FUTURE WORK.

In conclusion, both the ensemble techniques fail to produce a realistic human like chat experience. Perhaps training the base models on a much larger data set using using adversarial networks and reinforcement based training technique may significantly improve performance and generate non-generic human-like responses. Referring to the publications mentioned in section 2, and the ensemble technique, perhaps a model can be developed that has it's own persona cashed in memory and generates a persona for the user during a chat session, with a mechanism to store previous dialog states. Using this information, an ensemble model can be used to generate on-the-fly human-like responses. The diagram below shows a black box representation of such possible future approach to generate human-like chat bots:

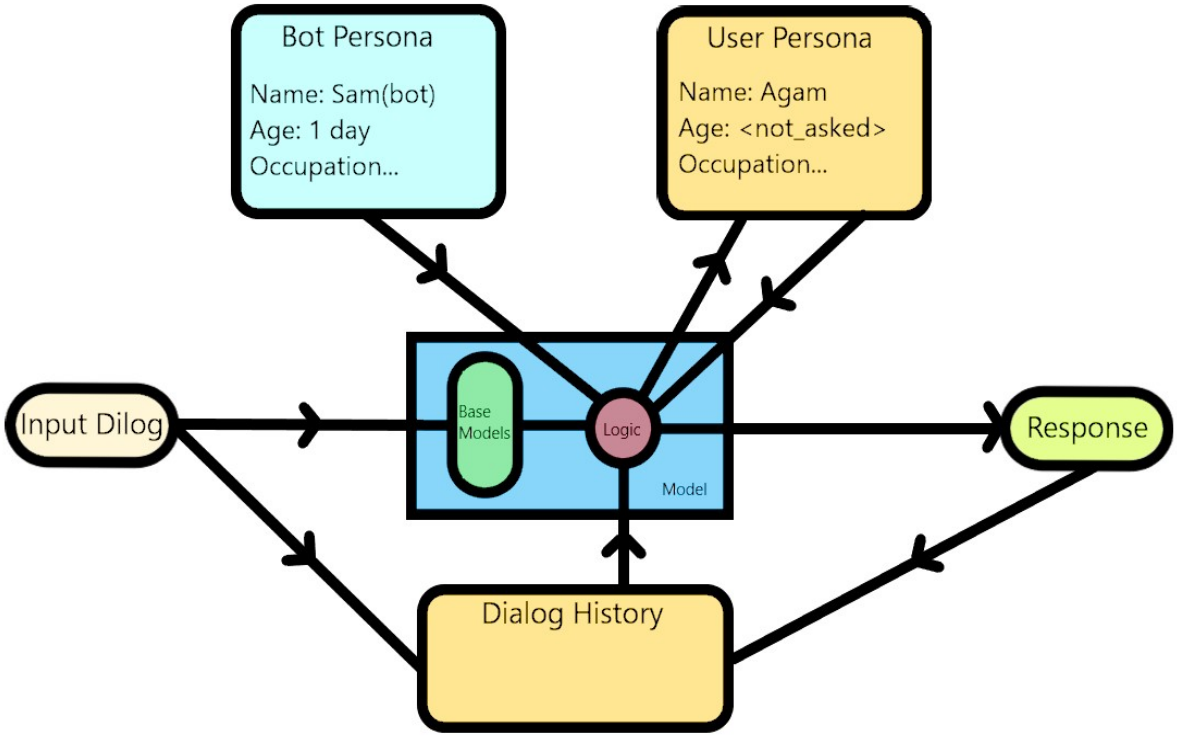

Fig.6.1. Proposed Black-box model to generate realistic human-like chat experience.

Here, only User Persona and Dialog history are allowed to be modified during a chat session i.e. at every new chat session, the model starts to build a profile for the user and logs the past dialogues to prevent repetition and deliver a more personalized chat experience. The Bot persona dictionary is passed at the beginning of the chat session and is not allowed to be modified. It contains the information necessary to generate a unique bot persona for dialog generation. The goal of the bot persona is to prevent inconsistent personas during a chat session and produce realistic and unique human-like responses. The base models and the algorithm logic will utilize a weighted F1 score technique discussed earlier to input to another algorithm that in combination with the personas and dialog history generate the next response. This simple technique should help address most issues faced in this project and help advance the field of language generation based chat-bots.