Capstone Project: Automation System for Email Response

Yuhan Shao
University of California, Los Angeles, Department of Computer Science
yuhan17@g.ucla.edu

ABSTRACT

The goal of this project is to propose an end-to-end Email Response Suggestion System. Different from most existing AI chatbot systems, the Email Response System would provide the user with multiple most relevant responses rather than only providing one reply. Accordingly, this project will investigate two approaches: (1) first construct a LSTM model, and then make inferences using the Beam Search algorithm. (2) As an alternative, a novel approach was employed as follows: First, the target texts were clustered and labelled, and then the input texts were used in conjunction with the labels to train the LSTM model. As part of the new approach, I chose epsilon to be 0.008 in the DBSCAN clustering algorithm, which generated 859 clusters as a result. Consequently, the new model achieves a high level of accuracy of 97%.
Keywords: LSTM, Beam Search, BLEU, GloVe, Annoy, DBSCAN

1 INTRODUCTION

In the information age, we always begin our day by checking and responding to our emails. To alleviate this cumbersome and tedious daily task, artificial intelligence has developed a deep learning technique based on natural language processing (NLP). It provides a solution that only requires users to click one tab to automatically generate replies to short and informal emails. An analysis of the millions of email-reply pairs in Google Gmail Smart Reply indicates that approximately one quarter of replies contain fewer than 20 tokens [4]. Therefore, it is sufficient to focus on short emails. This project involved the implementation of a sequence-to-sequence machine learning framework based on long short-term memory (LSTMs) [9]. In the baseline model, I loaded the pre-trained word embeddings (GloVe) [1] in order to generate the word embedding matrix. Using this matrix together with the training and validation datasets, I trained the model using teacher forcing [10]. Following that, I applied Beam Search [3] to infer a model from the test dataset, and evaluated the results of the inference using the BLUE score [6] metric.

The baseline model, however, does not guarantee the semantic quality of the responses since the predicted reply suggestions may have different semantic intents from the input message. In order to address this issue, I implemented a novel approach that involved clustering all similar target sentences in the dataset semantically, and then using the DBSCAN algorithm [2] to label each cluster uniquely. The clustering was conducted using Annoy (Approximate Nearest Neighbors Oh Yeah) [2] to index the input and target sequences and generate similarity matrices for DBSCAN clustering. As a final step, I trained the LSTM model with the input sentences and unique labels for the target clusters. The best prediction with an accuracy of around 97% was obtained with $\epsilon = 0.008$ after trying 50 different epsilon values between 0 and 0.03 evenly.

In the following sections, I will describe the related work, followed by the details of this project. This includes data preprocessing, the construction of models, and the inferences and evaluation of models. Finally, the paper will discuss future research opportunities and further discussion.

2 BACKGROUND AND RELATED WORK

Though Google has already published Smart Reply [4] in Gmail, there are no open-source implementations of the algorithm or implementations available for download. While a similar topic, Chatbot [1], has already been widely discussed. Email Response Systems can be considered modified versions of chatbots. Accordingly, a Chatbot application such as the one shown in Fig. 1 can conduct a 1-to-1 conversation via text or text-to-speech. As a result, the Email Response System may be viewed as a Chatbot that allows a 1-to-N conversation only through text. In the original implementation of the Response Generation Component in Fig. 1, it was a phrase-based Statistical Machine Translation (SMT) [7] system that responded to Twitter status updates. It was the first time that a method of generating full response predictions had been attempted.

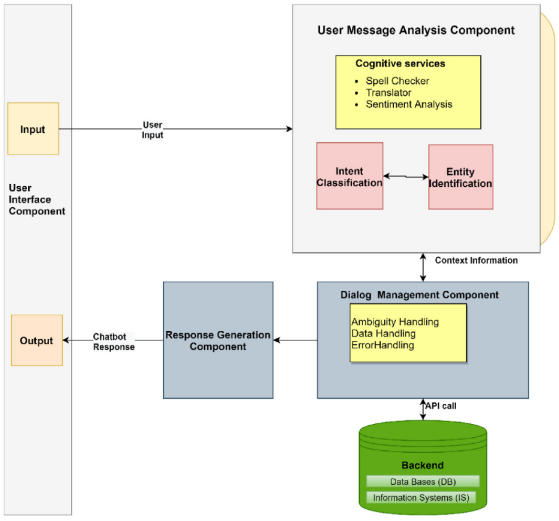

Figure 1: General Chatbot architecture

Recently, with the rise of AI and Machine Learning, people have developed neural networks [8] to solve this problem. In order to address the vanishing gradient problem that occurs with traditional recurrent neural networks, LSTMs were developed. Based on the

[1] GloVe: https://nlp.stanford.edu/projects/glove/
[2] Annoy: https://github.com/spotify/annoy/

architecture shown in Fig. 2, the input sequence is encoded into a fixed dimension vector by a multilayered LSTM. The target sequence is then decoded by another LSTM. Comparing this new approach to SMT, it achieves a higher BLEU score. As part of this project, I constructed a baseline model that is based on the LSTM encoder-decoder architecture.

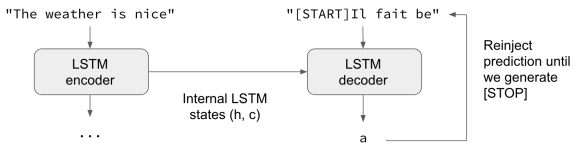

Figure 2: LSTM encoder and decoder

The baseline model, however, has a significant drawback. Regardless of whether the model is a vanilla RNN or one of its extensions, such as LSTMs or GRUs, the training procedure always uses maximum likelihood estimation. Maximum likelihood, however, would result in bias. For instance, if the previous word is used to predict a word, but this previous word is not found in the training data, the prediction becomes infeasible. Furthermore, if an error occurs early in the generation process, the generated sentence will diverge further [5]. In order to address this issue, Yu et al. (2016) developed a SeqGAN model [11]. Despite the large amount of work that has been done on GANs in the computer vision domain, a very small amount has been done in the natural language processing domain. It is because GANs are designed to operate on real-valued data, whereas text is represented as discrete tokens atomically. Consequently, I continued to use the LSTM as the new model, but rather than train a word-based LSTM, I constructed a response set-based LSTM model by first clustering the target texts.

3 DATA PREPROCESSING

To develop the baseline model, I used the Amazon Question/Answer Dataset [3]. There are 3 data files in this dataset, but only 'single_qna.csv' will be utilized in the baseline model. Over 20 categories of product-related questions and answers are included in this dataset file. It has 8 columns, but only 2 columns are used: 'Question' and 'Answer'. The new model makes use of the Amazon Topical Chat Dataset [4]. It consists of 8628 conversations and 184303 messages with different topics indicated by different conversation IDs. This dataset contains 3 columns: conversation_id, message, and sentiment. In this project, only the first 2 columns: 'conversation_id' and 'message', were used.

3.1 Data Cleaning

The data was cleaned by removing punctuation, numbers, HTML and emojis. All uppercase letters were replaced with their corresponding lowercase letters. Several common abbreviations have been extended. For example, 'canť' has been replaced with 'cannot'. Instead of using the entire dataset, I only consider input/question lengths less than 30 words, answer lengths less than 5, and target lengths less than 10 for the baseline dataset and the new model dataset, respectively. To capture broad and general responses to a wide range of queries entered by users, only short-length target texts were considered. Specifically, the '<bos>' and '<eos>' word tokens were added to the target messages to indicate the beginning ('<bos>') and the end ('<eos>') of the sentence respectively.

3.2 Data Transformation

As part of the cleanup of the Amazon Question/Answer Dataset, I created a column named 'QA' that stores the combination of questions and their corresponding answers. For both encoders and decoders, separate the dataset into the train, validation, and test sets. The Amazon Topical Chat Dataset was split into input text and reply text for each topic (conversation_id) after cleaning the data. Tokenization was performed on both models. Both the target and input sentences were kept fixed in order to apply batch training. As tokenization would assign each word a non-zero number, the post zeros were added to the tokenized sequences to make them equal in length. Specifically, I created dictionaries mappings between the index and the word in the baseline model. Fig. 3 illustrates the data preprocessing process for both the baseline model (pipeline on the upper part) and the new model (pipeline on the lower part).

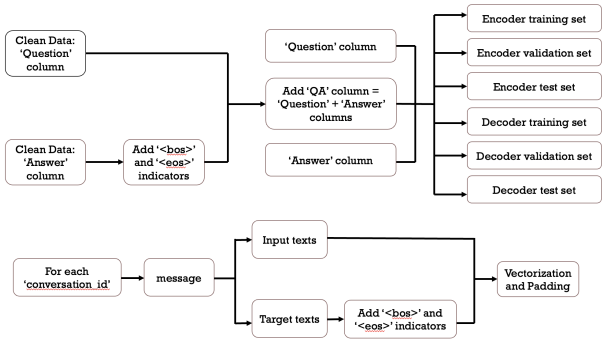

Figure 3: Data preprocessing for both baseline and new model

4 METHODOLOGY

4.1 Baseline Model: LSTM + Beam Search

With the Sequence-to-Sequence model described in [9], I implemented a basic Keras-based recurrent sequence-to-sequence model on a word-level. Fig. 4 illustrates the high-level architecture of the system. As a starting point, I loaded the pretrained word embedding GloVe (Global Vectors for Word Representation) [5]. In contrast to Google's Word2vec Pretrained Word Embedding, Stanford's GloVe Pretrained Word Embedding uses statistics from the entire dataset. The LSTM was trained using the training data $\vec{x} = (x, y)$, where input x represents the encoder_input batch and the decoder_input batch, and the target y represents the decoder_target batch. In the first step of the LSTM encoder, input x is transformed into two state vectors, the last LSTM state is kept and the outputs are discarded.

[3] Amazon QA Dataset: https://www.kaggle.com/datasets/praneshmukhopadhyay/amazon-questionanswer-dataset

[4] Topical Chat: https://www.kaggle.com/datasets/arnavsharmaas/chatbot-dataset-topical-chat

[5] GloVe: https://nlp.stanford.edu/projects/glove/

After that, the decoder LSTM is trained to convert the target y into the same sequence as the input x, but offset by one timestep in the future. This training process is known as 'teacher forcing'. To explain in more detail, after the encoder generates the state vectors, the decoder learns to generate targets {t+1, ...} given targets {1, ..., t}, based on the input sequences.

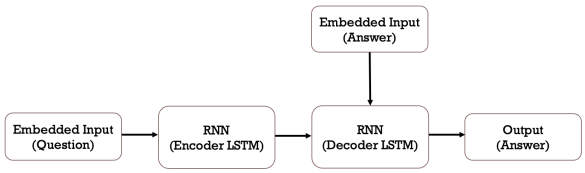

Figure 4: General Chatbot architecture

4.2 New Model: Clustering + LSTM

As previously mentioned, to eliminate the potential bias downsides in the text generation model using maximum likelihood, I constructed a response set based LSTM model by first clustering target texts.

4.2.1 Clustering. Let $I = \{i_1, i_2, ..., i_n\}$ and $T = \{t_1, t_2, ..., t_n\}$ represent the input texts set and target texts set, respectively. $\forall j \in [1, n]$, (i_j, t_j) is a valid conversation pair. Then, use keras Tokenizer to tokenize the text in sets I and T, encode and pad them by adding post 0s to make all the sequences in the same set have the same number of words. Denote the sets of sequences after the above process as the matrices $M_I \in \mathbb{R}^{n \times l_1}$ and $M_T \in \mathbb{R}^{n \times l_2}$ where:

$$l_1 = \max_{j=1,...,n} Number\_Of\_Words(i_j) \quad (1)$$

$$l_2 = \max_{j=1,...,n} Number\_Of\_Words(t_j) \quad (2)$$

After getting these 2 matrics: input sequences M_I and target sequences M_T, I first applied ANNOY to build a forest of 100 trees, then use this forest to find the k closet points to a given query point. Utilize this feature, I constructed the similarity matrices for both M_I and M_T. Denote the input sequences similarity matrices as $S_I \in \mathbb{R}^{n \times n}$ and target sequences similarity matrices as $S_T \in \mathbb{R}^{n \times n}$, where $S_I^{u,v}$ stores the similarity distance between u^{th} row of M_I and v^{th} row of M_I, which indicates the similarity between the u^{th} input text and v^{th} input text.

After getting the similarity matrics, I used DBSCAN clustering to uniquely label the target texts. The reason that I chose the DBSCAN algorithm is that, comparing to other clustering algorithms such as K-Means, DBSCAN does not need to know the number of clusters. Given a set of points, it clusters together the points in dense neighbourhoods and the algorithm would automatically converge to the a certain number of clusters on its own as shown in Fig. 5.

The parameter ϵ in DBSCAN clustering algorithm is the key which is the maximum distance between 2 points such that they can be

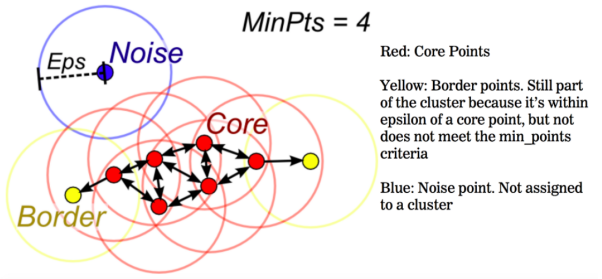

Figure 5: DBSCAN clustering

considered as the neighbour of each other. If ϵ is too large, all points would be included into 1 clustered. If ϵ is too small, all points would labeled as noises. An appropriate ϵ value should be chosen such that semantically close sentences get grouped together while unrelated sentences labeled as noises. In order to get the best ϵ value, I tried 50 values between 0 and 0.03 evenly and plot number of clusters and noises for each ϵ value as shown in Fig. 6.

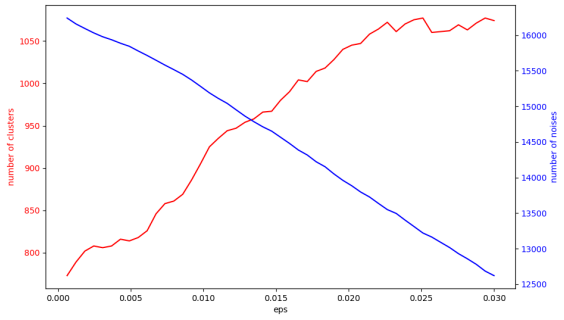

Figure 6: Epsilon number with number of clusters and noises

Therefore, choose $\epsilon = 0.008$, there are 859 clusters with 15513 noises, following table 1 shows some ϵ values with its correspond number of clusters and noises.

ϵ	#cluster	#noise
0.005	815	15831
0.008	859	15513
0.01	913	15243
0.015	969	14610
0.05	1113	10244

Table 1: Different eps in DBSCAN

4.2.2 LSTM. Using the clustering results from DBSCAN, encoded the categorical features as numeric arrays, then trained a single-layer LSTM model to predict the target cluster label based on the input text. An embedding layer is the first layer of the model, which takes in the integer representation of the word and converts it into

an N-dimensional dense vector representation. After this dense vector is fed into the next LSTM layer, a dropout layer is applied. At the last layer, there is a dense layer that is activated by a sigmoid function. Based on the input text, the whole architecture outputs a probability for each label. These predicted labels are the unique labels that DBSCAN assigns to the target texts. The whole architecture is shown in Fig. 7.

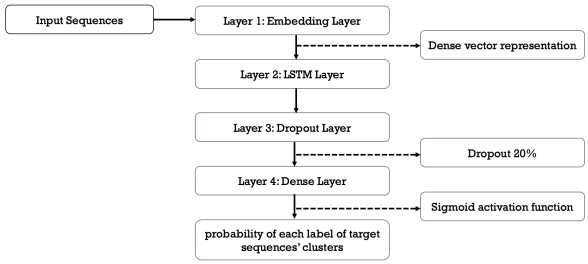

Figure 7: New model architecture

5 EXPERIMENTS

5.1 Baseline Model: LSTM + Beam Search

The training and validation loss of the baseline model is shown in Fig. 8. Text prediction is performed using beam search [3] in the inference mode. For the purpose of decoding unknown input sequences, it is first necessary to encode the input sequence into state vectors, and then use beam search to select the top K probable tokens and their corresponding state vectors. At the next time step, the top K tokens and state vectors are fed into the model. As a result, for each token, the next top K tokens will be generated, resulting in K^2 possible outputs. A probability score is calculated for all current K^2 final outputs, which is then sorted descendingly according to the score. Finally, only choose top K outputs from the K^2 results in this sorted array. Repeat this process until the '<eos>' indicator appears or the K beams reach their maximum length limit. Mathematically, given an input sentence t, for $\forall$ response $r \in$ response set R, where $|R| = K^2$, we want to find top K response such that:

$$s^* = \sum_{i=1,\ldots,k} \arg\max_{r_i \in R} P(r_i|t) \tag{3}$$

The baseline model was evaluated using the N-gram BLUE score (Bilingual Evaluation Understudy Score) [6]. Specifically, the N-gram BLUE score can be calculated as following:

$$BP = e^{\min(1-\frac{len(reference)}{len(prediction)},0)} \tag{4}$$

$$BLEU = BP \times \exp \sum_{k=1}^{N} w_k \log(p_k) \tag{5}$$

The BLEU score of baseline model choosing N = {1, 2, 3, 4} for N-gram is 0.2, 0.3, 0.35, 0.36, respectively.

5.2 New Model: Clustering + LSTM

Since the new model was constructed using Keras, I directly called the Keras callbacks history to access the training loss and accuracy.

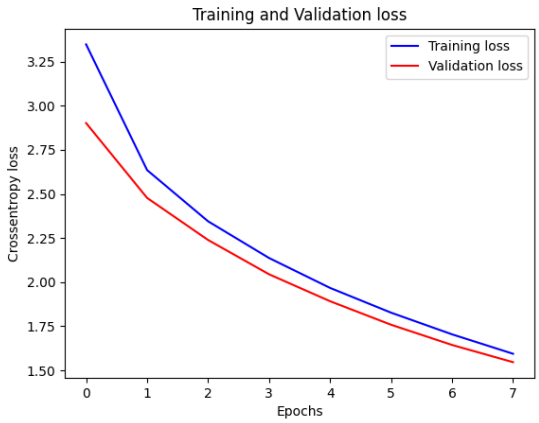

Figure 8: Baseline model loss with 8 epochs

Fig. 9 illustrates the relationship between training loss and training accuracy over 25 epochs. In addition, Fig. 10 provides screenshots of the sample inference examples in both the terminal output version and the web application version.

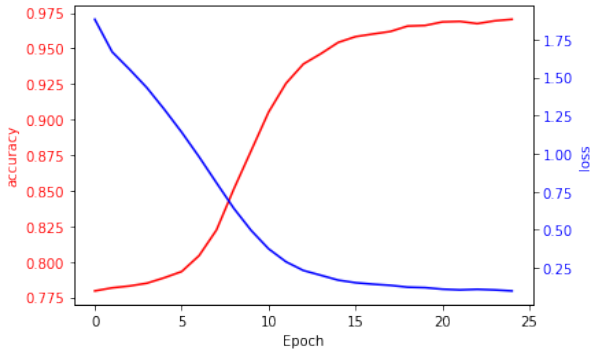

Figure 9: The new model loss and accuracy

6 FUTURE WORK AND DISCUSSION

In this project, I presented an end-to-end automation system for email response that was inspired by Google Gmail Smart Reply. In order to improve the quality of the response set, I proposed a novel approach that clusters the target texts semantically before feeding them into the LSTM model. In contrast to the baseline model which used beam search to find the top K score response candidates, the new model would generate more topic-related responses. As a downside, this novel approach would limit the creative potential of deep learning models. As shown in the left side of Fig. 10, when the query is 'do you like comic books', the answers are too general. None of the predictions include a 'comics' or 'book' related keyword. In this manner, neural networks are severely restricted in their ability to be creative. Future research should therefore take advantage of both the baseline model and the new model to come up with an innovative approach that increases both the accuracy of prediction and the creativity of prediction at the same time.

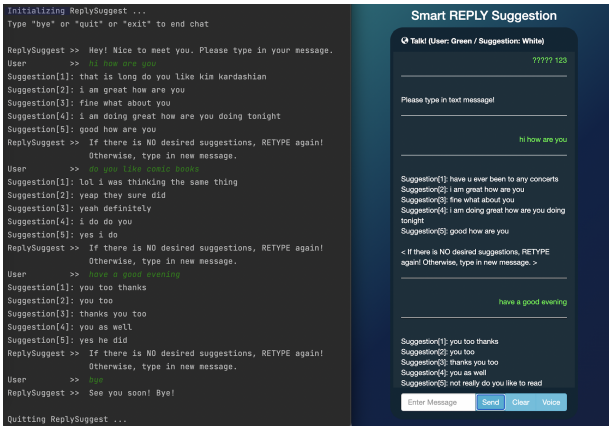

Figure 10: The preview of new model predictions

7 PROJECT GITHUB LINK:

https://github.com/syhAnna/CapstoneProject.git

REFERENCES

[1] Eleni Adamopoulou and Lefteris Moussiades. 2020. Chatbots: History, technology, and applications. *Machine Learning with Applications* 2 (2020), 100006. https://doi.org/10.1016/j.mlwa.2020.100006

[2] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (Portland, Oregon) *(KDD'96)*. AAAI Press, 226–231.

[3] Markus Freitag and Yaser Al-Onaizan. 2017. Beam Search Strategies for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics. https://doi.org/10.18653/v1/w17-3207

[4] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart Reply: Automated Response Suggestion for Email. https://doi.org/10.48550/ARXIV.1606.04870

[5] Prasad Kawthekar. 2017. Evaluating Generative Models for Text Generation.

[6] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (Philadelphia, Pennsylvania) *(ACL '02)*. Association for Computational Linguistics, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[7] Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-Driven Response Generation in Social Media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., 583–593. https://aclanthology.org/D11-1054

[8] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A Neural Network Approach to Context-Sensitive Generation of Conversational Responses. https://doi.org/10.48550/ARXIV.1506.06714

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

[10] Ronald J. Williams and David Zipser. 1989. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation* 1, 2 (1989), 270–280. https://doi.org/10.1162/neco.1989.1.2.270

[11] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. https://doi.org/10.48550/ARXIV.1609.05473