# Automatic Response Generation to Conversational Stimuli

Vishal Raj Dutta
vishal15115@iiitd.ac.in

Sanidhya Singal
sanidhya15085@iiitd.ac.in

## Abstract

*We present the interim project report for Group 36. In this project, we compare and contrast a few machine learning models that have been recently proposed and used in the automatic response generation to conversational stimuli. The project is based on generative models rather classification or clustering. Hence, sufficient effort has been involved in understanding and learning about them.*

## 1. Introduction

We′ve always been fascinated by how chatbots work and what makes them so realistic. In fact, response generation is a big challenge from the perspectives of machine learning and NLP, with new innovations always on the brink of unfolding.

Our project is based on automatic response generation to conversational stimuli, which seeks to make a machine comprehend conversations and respond accordingly. This has uses in several areas including e-mail, customer service and of course, chatbots.

More precisely we aim to play around with Neural Network frameworks and the features extracted from the data corpus (and possibly other frameworks), so as to achieve a well performing model. We also try using statistical machine translation models for the same purposes.

## 2. Related Work

The papers from which we′ve taken inspiration for the problem are [4], [6] and [9]. The underlying theme here, is to use bilingual translation models in response generation, with some modifications in how the data is input to the model. We′ve tried to explore both statistical and neural machine translation models in order to come up with a better model.

The neural conversational model proposed in [6] uses LSTM cells in an RNN to remove the issue of vanishing gradients, and hence has better performance.

Google′s Smart Reply [4] uses the neural frameworks proposed in [6], but with various add-on mechanisms such as the Triggering mechanism which also detects whether or not to even create responses to a given mail. It also uses a semi-supervised graph learning approach to deliver better responses. Additionally, there are methods suggested for increasing diversity in the response subspace created.

Also, [6] uses phrase-based models for Statistical Machine Translation, which is different from the Neural Machine Translation models explored above, but generates appreciable response results.
Some other relevant works are: [5], [1], [8], [3], [2].

## 3. Data set and Evaluation

### 3.1. Data set

Data set currently in use is the **Cornell Movie Dialog Corpus**, with over $6,00,000$ lines of dialogue. As of now we have $5000$ samples in the training set, $400$ samples in validation set for tuning hyperparams, and $400$ samples in the test set, all from randomly chosen conversation pairs. We′ve also cleaned and pre-processed the **Enron mail corpus** into stimulus-response pairs, for future use.

### 3.2. Feature Extraction

Given an input sentence of length $L$, it is first padded with pad tokens, till maximum length ($maxL$) is reached. Our feature vector is then a 1-hot styled vector of all the words in our vocabulary ($\sim 4000$) (with rare words ($< t\%$) replaced by UNK). Each word has a corresponding integer ID; thus, they are first converted to their integer forms and then the above feature vector is created.
e.g.: if our vocabulary is $\{'$cat$'$, $'$i$'$, $'$am$'$, UNK, PAD$\}$ $\rightarrow \{1, 2, 3, 4, 5\}$ and $maxL = 5$, then $\{'$i am boy$'\}$ is converted to $\{'$i am boy PAD PAD$'\}$, and its word vector is: $\{(01000)(00100)(00010)(00001)(00001)\}^T$.
All training stimulus pairs are converted into the format above, and are fed into the encoder model, and all corresponding response pairs are converted into the format above, and are fed into the decoder model.

### 3.3. Evaluation

We choose **Perplexity** to evaluate the model′s training performance. Perplexity is a measure of confidence of predicting the next word in the response given. A perplexity of

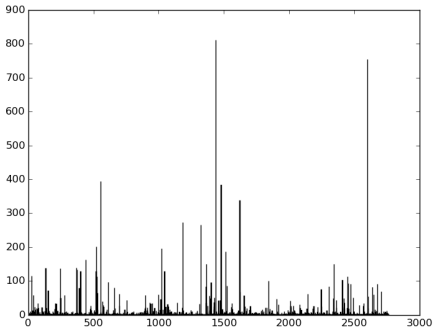


Figure 1. Distribution of words and their corresponding frequencies in the corpus

value $k$ means that there are around $k$ possible candidates while predicting the next word. Obviously a better trained model will have lesser value of perplexity as it can better filter appropriate candidates and hence, is more confident about its prediction.
The perplexity on a set of $N$ test samples is computed using the following formula:

$$P_r = exp(-\frac{1}{W}\sum_{i=1}^{N}\ln(\hat{P}(r_1^i,...,r_m^i|o_1^i,...,o_n^i))$$

where $W$ is the total number of words in all $N$ samples, $\hat{P}$ is the learned distribution and $r^i$ and $o^i$ are the $i$-th response and original message respectively. Note that in the equation above only response terms are factored into $P_r$. Loss function used for per-epoch evaluation in our model is **cross-entropy**.

## 4. Analysis and Progress

The distribution of words and their corresponding frequencies has been shown in Fig. 1.

### 4.1. Models Implemented

So far, we have designed and implemented the models for **RNN using LSTM**, and **RNN using GRU** in **Keras** with **Tensorflow** as back-end. We did conduct several iterations, of both models, but as of now have gotten unsatisfactory results. Neural networks are of course computationally expensive, and we were handicapped in that segment. Nevertheless, we wish to obtain means to better machines, and train our model better in the near future. For statistical machine translation, we′re ongoing with the implementation (using **NLTK**) and hope to get it ready for evaluations soon.

### 4.2. Challenges faced

One of the primary challenges faced in our project is that the neural network uses very common words as *safe bets*. For example, since *i* or *you* are seen very frequently in the data (Refer Fig. 1.), the network thinks its safe to assume strongly that the next possible word is either *you* or *i*. So, for a small number of epochs ($\sim 100$) on a training set of $5000$ samples, most common responses are *i i i i i i* or *i you you i i i*, etc. Hence, we require some kind of frequency normalization, to make this notion of *safe bets* obsolete. We also wish to increase number of iterations to a minimum of $1000$ iterations if not more. Also, for now we have used naive tokenization techniques (splitting by spaces), which may lead to redundancies in our vocabulary, e.g.: $said$ and $said,$. To resolve this issue, we use NLTK's word tokenizer, which tokenizes our data in a better way.

### 4.3. Design Choices

We initially started with a simple RNN model, but later changed it to RNN with LSTM, since we became aware of the notion of vanishing gradients upon reading [9]. We′re also exploring RNN with GRU to compare and contrast the two. We are currently studying the effect of epochs and number of latent dimensions in both these models. We also plan on making a character-by-character translation model, and evalutate it against our current word-by-word model.

## 5. Results

For our current training model, the best result yet is a perplexity value of $144.707$ with number of epochs $= 100$ and latent dimensions $= 2$. The model is an RNN-LSTM and features are word vectors described above.

```
('Input sentence:', ' well i thought we
d start with pronunciation if that s
okay with you.'
('Decoded sentence:', ' i i i i i i i i
i i i i i i i i i i i i i')

('Input sentence:', ' not the hacking
and gagging and spitting part. please.')
('Decoded sentence:', ' i i i i i i i i
i i i i i i i i i i i i i')

('Input sentence:', ' you re asking me
out. that s so cute. what s your name
again?')
('Decoded sentence:', ' s s s s s s s
s s s s s s s s s s s s s')
```

### 5.1. Inferences

As described in challenges, we see common tokens appearing all the time. This probably has something to do with the notion of *safe bets* as discussed earlier. Since the perplexity is high, we can admit that our model is yet to train better. Possibly more hidden layers, more number of epochs

or more training data can improve these statistics. There is a huge gap of performance between our model and Google′s Smart Reply [4] which has a perplexity of 17.1. Hence, we need to find ways to better our model, for now.

## 6. Future Work

### 6.1. Learning techniques used

We aim to tweak our model in terms of the neural cells used, and if possible come up with a better model.

### 6.2. Modifications in dataset choice

We had initially planned to use the Enron Mail corpus as our data, but have now shifted to the Cornell Movie–Dialogs Corpus, obtained from https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html. This was due to our shift from a mail-response domain to a chat-response domain, mainly because the Cornell Dialogs corpus was more accustomed to everyday conversation, and had a barrage of conversational topics, but the Enron mail corpus was business oriented, which was not a domain we were comfortable in.

### 6.3. Change in evaluation metrics

We are currently using perplexity as an evaluation metric for our models, as it is a widely followed metric in the NLP domain. The previous evaluation metrics suggested were classification related, which cannot be used as our problem lies in the generative models domain.

### 6.4. Analyses

We aim to:

1. Compare and contrast word-by-word vs. character-by-character translations of data
2. Study the effect of word embeddings on the performance of the model
3. Study the effect of reversing the input word sequence first before converting to vector form, so as to increase performance [7].

Mainly, we also wish to come up with a new model to better incorporate the challenges that we have faced thereof, and implement an IR-baseline also.

### 6.5. Individual team member roles

Sanidhya: Analysis on RNN-GRU, Statistical machine translation Model, Developing new model
Vishal: Analysis on RNN-LSTM, IR-Model (nearest-neighbours), Developing new model

## References


[1] K. Cho, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoderdecoder for statistical machine translation, 2014. https://arxiv.org/pdf/1406.1078.pdf.

[2] G. Corrado. Computer, respond to this email. https://research.googleblog.com/2015/11/computer-respond-to-this-email.html.

[3] Deeplearning4j. A beginners guide to recurrent networks and lstms. https://deeplearning4j.org/lstm.html.

[4] A. Kannan, K. Kurach, S. Ravi, T. Kaufman, B. Miklos, G. Corrado, A. Tomkins, L. Lukacs, M. Ganea, P. Young, and V. Ramavajjala. Smart reply: Automated response suggestion for email, 2016. https://research.google.com/pubs/pub45189.html.

[5] K. Nishimura, H. Kawanami, H. Saruwatari, and K. Shikano. Investigation of statistical machine translation applied to answer generation for a speech-oriented guidance system, 2011. http://www.apsipa.org/proceedings_2011/pdf/APSIPA066.pdf.

[6] A. Ritter, C. Cherry, and W. B. Dolan. Data-driven response generation in social media, 2011. http://aclweb.org/anthology/D11-1054.

[7] T. Tran. Creating a language translation model using sequence to sequence learning approach. https://chunml.github.io/ChunML.github.io/project/Sequence-To-Sequence/.

[8] T. Tran. Creating a text generator using recurrent neural network. https://chunml.github.io/ChunML.github.io/project/Creating-Text-Generator-Using-Recurrent-Neural-Networ

[9] O. Vinyals and Q. V. Le. A neural conversational model, 2015. https://arxiv.org/pdf/1506.05869.pdf.