Awkwardly: A Response Suggester
CS224N Final Project

Quinlan Jung
Department of Computer Science
Stanford University
quinlanj@stanford.edu

Kai-Chieh Huang
Department of Electrical Engineering
Stanford University
kaichieh@stanford.edu

Abstract

Replying to emails can be a daunting task especially if users are bombarded with hundreds of emails per day, or if they struggle with constructing well-formed socially acceptable replies. We present *Awkwardly*, a novel response suggester that generates top replies shown to a user in real-time. These short responses can be selected as a reply in a chat or email context. In this paper, we introduce methods to generate candidate responses. First, we generate a large pool of responses using various seq2seq LSTM neural network architectures. Next, we group and rank the responses using semantic intent clustering. These top ranked responses are relevant and diverse, and they are the ones that are ultimately shown to the user.

1 Introduction

Replying to emails is a difficult and time consuming task. The average person receives 97 business emails and 88 personal emails per day. [1] Time spent on emails is significant, as the average US employee spends 28% of their work week replying to emails. [2]. Clearly, there is a need to reduce the processing time required to generate a well-formed response to emails a user must reply to.

We present *Awkwardly*, a response suggester. *Awkwardly* is a deep learning NLP model that can generate text responses from input such as email, chat or dialog. The model will take in a prompt such as 'Hey, what day are you free?' and suggest replies such as 'Tomorrow works for me' and 'Thank you but now is not a good time'. If the user feels that one of the suggested replies is appropriate, they select one of them, modify it as needed and send it to the recipient.

Given an input, we gather a set of similar input ranked in terms of semantic intent. Then we feed our set through a seq2seq multilayer LSTM to generate a large pool of responses. We also optimize our LSTM by adding an attention mechanism to allow the decoder more direct access to the input. Our encoder LSTM cell receives the conversational prompt in the form of Glove vectors. Our decoder receives one-hot vectors of the user's reply that maps to corresponding pre-trained embedding during training, and one-hot vectors of it's own output during testing. The output of our network is a stream of one-hot vectors, representing a response.

Next we take our set of responses to group and rank them using semantic intent clustering [5] and the EXPANDER algorithm. [6] Once the responses have been clustered, we filter out the ones with low readability. We only keep responses that have a Flesch-Kincaid score above 60, which ensures that each response is written in plain English and is understood by 13 to 15 year old students. The highest ranked responses in each cluster is presented to the user.

2 Related Work

Over the past year, neural machine translation (NMT) has become the state of the art implementation for language translation. [7] The first encoder-decoder model using two basic RNN cells was presented by Cho et al. [8] Multi-layer cells have been successfully used in encoder-decoder models too. [9] The advantages of a seq2seq encoder-decoder model over a general feed forward neural network is that it can remember input from previous time-steps. Also it conditions the generated words on both input and previously generated outputs to predict a meaningful response.

In the above NMT implementations, every input has to be encoded into a fixed-size state vector, as that is the only thing passed to the decoder. To allow the decoder more direct access to the input, an attention mechanism was introduced by Bahdanau et al., which allows the decoder to peek into the input at every decoding step and pay attention to some key words. [10]

In addition to the wide success NMT has seen in language translation, it is also adopted for context-answer type problems. Chatbots have been made using the encoder-decoder model, using the messages leading up to the user's reply as input to the encoder and the user's reply as input to the decoder. [11] Email suggesters have also been built on such models by Kannan et al, grouping replies using semantic intent clustering and choosing the most appropriate one from each cluster. [5]

3 Approach

Our high level algorithm to generate a set of suggested responses consists of the following steps:

1. Given an input, gather a set of similar inputs.
2. Feed the input set through a seq2seq neural network to generate a set of responses.
3. Cluster and rank responses.
4. Perform additional filtering to ensure baseline readability.
5. Present the highest ranked response in each cluster to the user.

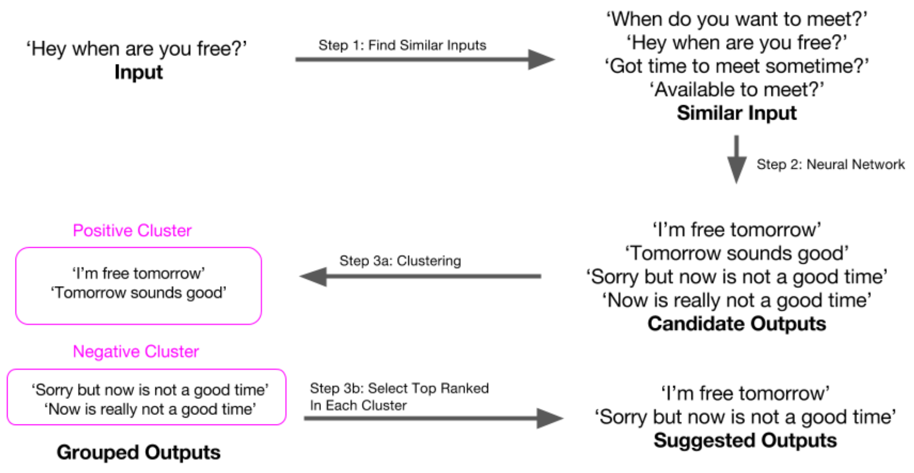

Figure 1: A flowchart of our high level algorithm.

We define the term 'context' to mean the part of the conversation leading up to the user reply and 'response' to mean the reply the user responded with based on the context.

3.1 Baseline Seq2Seq Model

For our baseline neural network, we closely follow the seq2seq model presented by Cho et al. [8] The inputs to the encoder are one-hot vectors of the context, and the inputs to the decoder are one-

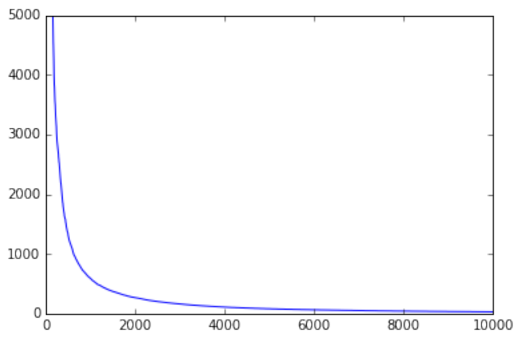

Figure 2: # Words vs Word Frequency in the Twitter Corpus [14]

hot vectors of the response during training, and one-hot vectors of its own output during testing. However, instead of using a basic RNN cell, we use a basic LSTM cell for both the encoder and decoder, and instead of using a Adadelta optimizer, we use the Adam optimizer. We also add an additional Dropout wrapper around each cell, setting the dropout rate to 0.5. We added the Dropout wrapper to prevent overfitting, with the suggested dropout rate being 50% as it has often been found to result in the maximum amount of regularization. [13]

For our data, we use Marsan Ma's curated twitter corpus of $800,000$ context-responses. [12] We set the context to be the original tweet, and the response to be the user's reply to said tweet. While Cho et al's dataset is significantly larger (61M words), our dataset has 9M words.

We perform basic tokenization of each word in our dataset. We use an initial vocabulary of the $|V| = 6000$ most frequent words seen in the corpus. All other words get replaced with an *UNK* token. In order to optimize for training time and memory usage, we choose a vocabulary size of $|V| = 6000$, which means 3% of our tokens are *UNK*. We ensure we do not exceed the recommended threshold of 5% unknown tokens. [15]

We trained our baseline for $n = 40000$ epochs. During test time, we forbid the decoder from outputting the *UNK* token. If the *UNK* token is the token predicted with highest probability by $\hat{y}$, we choose the token with second highest probability instead.

After $n = 40000$ epochs, we see that our model outputs grammatically correct responses, but the variation is poor ($> 90\%$ are variations of generic responses such as *'I think you are not racist*', *'I dont know'*, *'I love you'* and *'The media is a disgrace'*).

3.2 Additional Seq2Seq Optimizations

3.2.1 Glove Inputs

We use pretrained Glove vectors on a Wikipedia corpus as input to the encoder instead of the one-hot vectors we used in our baseline. We initially trained our neural network on embedding dimensions of $D = 50$, but increased it to $D = 300$ due to the low variation of responses (the model kept outputting *'I love you'* and *'I dont know'* for most prompts at $D = 50$).

3.2.2 Larger Vocabulary Size

We increased the vocabulary size to $|V| = 100000$. Initially, we were skeptical performing this optimization would work, but discovered that having a $< 1\%$ *UNK* tokens in the training set greatly increased the quality of responses. (ie) Instead of outputting *'I love you'* in response to the prompt *'Its my birthday!'*, we received the higher quality output of *'Happy birthday!'*.

3.2.3 Multilayer LSTM

Instead of having each cell be a Basic LSTM, we use a 3-layer stacked LSTM. It is recommended that going from a basic LSTM cell to a stacked LSTM can help to obtain better results by allowing for greater model complexity. [17]

3.2.4 Attention Mechanism

We further enhanced our seq2seq model with the attention mechanism presented by Bahdanau et al to allow the decoder focus on certain ranges in the input sentence. [10] By introducing an adaptive weight and calculating a weighted hidden state using the hidden states from all time steps, we can provide the seq2seq model with additional information and allow the model to pay attention on particular words inside the input context.

3.3 Baseline Response Clustering and Ranking

In order to cluster and rank our responses (Step 3 of our high level algorithm), we perform k-means clustering and get the nearest neighbours to each centroid. [1]

To group inputs, we perform k-means clustering on vectors derived from our inputs. These vectors were an average of the Glove representation of each input token. We perform k-means clustering on Glove vectors of $D = 300$ dimensions.

To ranking each group, we find the nearest neighbour measured in Euclidean distance and output the nearest vector to each centroid returned by k-means. To find the nearest neighbour, we construct a k-dimensional tree from our response set. [16] Then, we output the nearest vectors as our highest ranked response in each cluster.

Our baseline results did not yield the results we intended. Of $n = 200$ iterations of 2-means clustering, the closest vectors to the centroids were always *'I love you'* and *'I love you too'* with 2.5% of the iterations returning *'I dont know'*. One possible explanation for these unexpected results is that *'I love you'* and *'I love you too'* have the most average Glove vectors in $D = 300$ dimensional space.

3.4 Semantic Intent Clustering

We closely follow the implementation of grouping and ranking candidate responses using the semantic intent clustering and EXPANDER algorithm described by Kannan et al. [5] Semantic intent clustering allows responses to be grouped by their intent (ie) responses in the appreciative group include *'thank you'* and *'thanks so much'*, whereas responses in the affectionate group include *'i love you'* and *'i love you too'*. In this semi-supervised algorithm, a few manually seeded groups are created for different categories (ie) the appreciative and affectionate group of messages mentioned previously.

A base graph is then constructed, with the manually seeded messages comprising of nodes V_R. For each message, a set of lexical features (ngrams and skip-grams of n up to 3), are created and comprise of feature nodes V_F. Edges are created between a pair of nodes (u, v) where $u \in V_R$ and $v \in V_F$ if v belongs to the feature set for response u.

The constructed graph captures relationships between similar responses via the feature nodes. The semantic intent information is then propagated from the manually labeled examples through the graph using the EXPANDER framework. [6] For each message, the top scoring output label is the semantic intent it gets clustered into. The message with the highest score in each cluster is returned to the user.

[1] We perform a similar algorithm in Step 1, where we take the centroid from 1-means clustering and find the N nearest neighbours to get our input set.

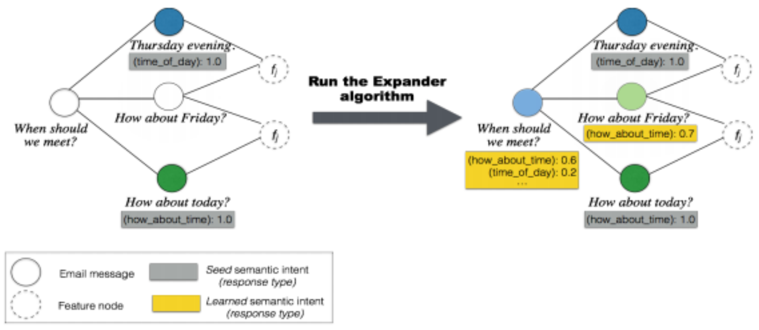

Figure 3: Semantic clustering using the EXPANDER algorithm [5]

3.5 Flesch-Kincaid Filter

The Flesch-Kincaid score indicates how difficult a passage in English is to understand by examining a response's sentence, word and syllable ratio. It is used extensively in the field of education. A score above 60 is deemed to be plain english and easily understood by 13 to 15 year old students. Anything below this score is considered to be difficult to understand. [4] Thus, we filter out messages that has a score below 60 in Step 6 of our high level algorithm description .

In order to determine the number of syllables in each word, we use cmudict, a pronouncing dictionary for North American english words. Each phoneme that contains a stress marking has a value of 0, 1 or 2. We count the stress markers, which effectively gives the number of syllables.

$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$$

Figure 4: The Flesch-Kincaid score [5]

4 Evaluation

In this section, we present two evaluation metrics for quality assessment of our response suggester model. Following the trend of recent seq2seq model papers, we adopt Perplexity as our first evaluation metric. Then we propose two ratios: Unique Response Percentage (URP), and Meaningful Response Percentage (MRP) as a unique metric for evaluating NLP response systems. The details of these two evaluation metric are presented in the following subsections.

4.1 Perplexity

As suggested in [5], perplexity is used to measure how well the model has fit the data. Generally, the perplexity is calculated as an inverse of the normalized likelihood for which the model fits a response. As a result, a lower perplexity means the model assigns higher likelihood to the test responses and is better at predicting responses. For example, for the ideal scenario of perplexity equal to 1, we would expect the model to predict exactly what should be the next word. To calculate the perplexity of a model trained on a set of N test samples, we compute the following formula:

$$PP = exp(-\frac{1}{N}\sum_{i=1}^{N}\bar{\ln}(\hat{P}(r_1^i, ..., r_m^i \| c_1^i, ..., c_n^i))$$

where N is the number of training samples, and r is the i-th response and c is the i-th input context message. $\hat{P}$ is the likelihood estimation of the i-th response given the i-th input context message. Finally, let $\bar{\ln}$ denote the log likelihood of one training example normalized by the number of time steps of the sentence. We thus represent the perplexity of a model by the average perplexity over all training examples. The perplexity comparison of our final response suggester model and the baseline is presented in Table 2.

4.2 URP & MRP

It is common for seq2seq models using deep learning to generate similar responses such as ”I don’t know”, ”I love you”, or ”Thank you” when encountering unfamiliar input sentences. We expect a competent seq2seq model to generate as many unique responses as possible based on different input sentences. In addition, the model is not guaranteed to generate meaningful or grammatically correct sentences. A good language model should generate a high percentage of grammatically correct and meaningful responses. Thus, to further understand the performance of different response suggestion models, we introduce two additional metrics: Unique Response Percentage (URP), and Meaningful Response Percentage (MRP) to evaluate the model on these two aspects.

To evaluate the model’s ability to generating unique responses, we feed the model with 100 input sentences and calculate the percentage of unique responses it generates. We refer to this as the Unique Response Percentage (URP). Similarly, we calculate the percentage of grammatically correct and meaningful responses generated by the model. We refer to this as the Meaningful Response Percentage (MRP). Using these two scores, we can learn the diversity and robustness of a given seq2seq model. The comparison of URP and MRP between the baseline and our final response suggester model is also presented in Table 1.

Model	Perplexity	URP	MRP
Baseline	29.6659	22%	14%
Attention seq2seq	22.5615	74%	38%

Table 1: Performance comparison

5 Results

From Table 1 in the previous section, we can see that our final attention optimized seq2seq outperforms the baseline implementation by all evaluation metrics. By adopting the attention optimized seq2seq model, we were able to decrease the perplexity of our model to 22.5615. Furthermore, since we explicitly forbid the *UNK* token by selecting the second most probable word as output and also increase the vocabulary size to 8000, we successfully increase the URP from 22% to 74%. Combining the benefit of the attention mechanism and forbidding the *UNK* token, our final response suggester model also has a better MRP where more meaningful and grammatically correct responses are generated.

Some good examples curated from the baseline model responses are presented in Table 2 and some bad examples are shown in Table 3. Even with the baseline model, it is generating some proper responses. However, the model generates a lot of *UNK* tokens most of the time due to the small vocabulary size we use. Also, without the attention mechanism, the model is prone to responses with some *UNK* tokens if there are also *UNK* tokens in the input context. This is due to the baseline model’s lack of ability to down-weight the *UNK* tokens in the input context and focus on the more meaningful part of the sentence.

Input context	Response
how you been though bro i see youre doing some clothes stuff	i miss you so much i love you so much
ive gotten a few pics from the wedding	this is so cute
so many all stars in one story nice	i love it
happy birthday have a very lit birthday	thank you
favorite christmas song	this is the best
i hate that word but hell yeah lol	lol i was just thinking about it
memes are no joke in this group chat	yes i know right
thank you	youre welcome
i think its over bud	i know i was just thinking about it
heres the customer care link for you eh	thanks for the shout out

Table 2: Example of good responses from baseline model

Input context	Response
describe yourself in three fictional characters	unk unk unk
what trump wants to do hillary has already done	unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk
lol beat that ass	lol he was unk
you can watch the whole thing on youtube	i think the unk is unk
new poll gives clinton debate win now up to 5 real polls all telling the same story	hillary is a unk unk
hey and im not gonna keep editing this shit for free	im going to be a unk

Table 3: Example of bad responses from baseline model

In Table 4 we present examples of good responses curated from the attention seq2seq model as well as some bad responses in Table 5. As discussed previously, the model is generating much less *UNK* tokens, as we explicitly forbid it by selecting the second most probable word as output and also increase the vocabulary size to 8000. As a result, the bad response generated by the attention optimized seq2seq model is mostly repeated phrases or grammatically incorrect sentences.

Moreover, proper responses of the attention seq2seq model have a more explicit relationship to the input sentences. For example, when given the input context of *what did everybody think of last nights episode*, it responds with *i was watching it*. Here, ”episode” and ”watching” have a much more explicit relationship compared to a more generic response such as *i love it*, which is an appropriate response to many different input contexts. This demonstrates our final response suggester's capability of paying attention to particular words in the input context and generating responses having an explicit relationship.

Input context	Response
have you ever seen donald trump laugh via	i think he was a racist
so many all stars in one story nice	i love it
from a fan looking for someone to build a good top quality business website thanks please comment below	we can help you in design and development of your website at reasonable price visit our portfolio
happy birthday craig	thank you
what did everybody think of last nights episode	i was watching it
i wouldnt participate in another debate rudy giuliani says	i am not sure
wishing you wanted me as bad as i want you	i love you too
i played everything on that game	i was watching the game
bet that you know we gonna turn up	i dont know what i was doing

Table 4: Example of good responses from attention seq2seq model

Input context	Response
lol damn thats harsh	i dont know what you want to do
definitely not opposed to this	i think i was thinking to the
youre hired	i was a to
these three made it happen	i think i was thinking
definitely excellent gif choice btw how could i say no	i think i think i think i think i think i think i think i think
join us ill figure out a spot later tonight or tomorrow	i is

Table 5: Example of bad responses from attention seq2seq model

6 Conclusion

In our final implementation, much of our work was constrained by the available memory and training speed of our computing instance. When training on the attention optimized seq2seq model, the GPU runs out of memory when using a vocabulary set of over 8000 words due to the increase of parameters used for the attention mechanism. When using a small vocabulary size, a large portion of the training input context and responses contains the unknown token. This can restrain the model's ability to learn important information from the training examples, since the context becomes hard to understand, even for humans. As a consequence, we expect a better model to be trained by using a larger vocabulary set.

As discussed in previous sections, the resulting model generates promising responses with twitter-like input sentences while lacking the ability to follow conversational inputs such as ”How are you?” or ”Hi my name is John, what's your name?”. The model has become overfitted to twitter-like inputs. In order to build a better model, one possible approach is to train the model on more diverse training examples. It would be interesting to concatenate the Cornell Movie-Dialogs corpus in addition to the Twitter corpus to train a better model, provided one has sufficient time.

Other future directions include exploring the technique used in other language models such as dialog modeling. Since the nature of dialog modeling and email/text replying is essentially the same, many interesting concepts discovered in dialog modeling literature also apply to response suggestion models. In particular, the personalized information capture mechanism described in [18] can be utilized to learn the information of a particular user. This way, the model can become more consistent in generating suggested responses that incorporate the user's personal information. For example, when given an input context such as ”Where do you live”, the model should consistently give a response suggestion of the user's address.

References

[1] http://www.radicati.com/wp/wp-content/uploads/2015/02/Email-Statistics-Report-2015-2019-Executive-Summary.pdf

[2] http://www.mckinsey.com/industries/high-tech/our-insights/the-social-economy

[3] Wuchty, Stefan and Brian Uzzi. ”Human Communication Dynamics: A Study of the Agreement between Self-reported and Email Derived Social Networks,”PLoS ONE 6(11): e26972. doi:10.1371/journal.pone.0026972, 2011.

[4] Kincaid, J.P., Fishburne, R.P., Rogers, R.L., & Chissom, B.S. (1975). Derivation of new readability formulas (automated readability index, fog count, and flesch reading ease formula) for Navy enlisted personnel. Research Branch Report 875. Chief of Naval Technical Training: Naval Air Station Memphis.

[5] Smart Reply: Automated Response Suggestion for Email, Anjuli Kannan , Karol Kurach, Sujith Ravi, Tobias Kaufman, Balint Miklos, Greg Corrado, Andrew Tomkins,Laszlo Lukacs, Marina Ganea, Peter Young and Vivek Ramavajjala, 2016, Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD). URL = https://arxiv.org/pdf/1606.04870v1.pdf

[6] Large Scale Distributed Semi-Supervised Learning Using Streaming Approximation, https://arxiv.org/pdf/1512.01752v2.pdf

[7] http://www.androidpolice.com/2017/03/06/google-translate-now-uses-neural-machine-translation-languages/

[8] Learning Phrase Representations using RNN EncoderDecoder for Statistical Machine Translation https://arxiv.org/pdf/1406.1078.pdf

[9] Sequence to Sequence Learning with Neural Networks https://arxiv.org/pdf/1409.3215.pdf

[10] Learning Phrase Representations using RNN EncoderDecoder for Statistical Machine Translation https://arxiv.org/pdf/1406.1078.pdf

[11] A Neural Conversational Model, Oriol Vinyals and Quoc V. Le, CoRR, abs/1506.05869, 2015, URL =http://arxiv.org/abs/1506.05869

[12] https://github.com/Marsan-Ma/twitter_scraper

[13] Understanding Dropout, http://papers.nips.cc/paper/4878-understanding-dropout.pdf

[14] http://suriyadeepan.github.io/2016-12-31-practical-seq2seq/

[15] Unsupervised Lexicon-Based Resolution of Unknown Words for Full Morphological Analysis, http://www.aclweb.org/anthology/P08-1083

[16] Bentley, J. L. (1975). ”Multidimensional binary search trees used for associative searching”. Communications of the ACM. 18 (9): 509. doi:10.1145/361002.361007.

[17] https://deeplearning4j.org/lstm

[18] A Persona-Based Neural Conversation Model, Jiwei Li, Michel Galley , Chris Brockett, Jianfeng Gao and Bill Dolan, 2016, abs/1603.06155, CoRR, URL = http://arxiv.org/abs/1603.06155