International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 08 | Aug -2017 www.irjet.net p-ISSN: 2395-0072

LSTM Model for Semantic clustering of user-generated content using AI Geared to wearable Device

Dr. T.Suresh[1], Dr. K.T. Meena Abarna[2]

[1,2] *Assistant Professor, Dept. of Computer Science and Engineering, Annamalai University, Tamilnadu, India.*

Abstract - *In this paper we propose and investigate a novel end-to-end method for automatically generating short message responses, called Smart Reply. It generates semantically diverse suggestions that can be used as complete message responses with just one tap on wearable. The system is currently used in Inbox by Message and is responsible for assisting with 10% of all wearable responses. It is designed to work at very high throughput and process hundreds of millions of messages daily. The system exploits state-of-the-art, large-scale deep learning.*

We describe the architecture of the system as well as the challenges that we faced while building it, like response diversity and scalability. We also introduce a new method for semantic clustering of user-generated content that requires only a modest amount of explicitly labeled data.

Key Words: Message, LSTM, Deep Learning, Clustering, Semantics.

1. INTRODUCTION

Message is one of the most popular modes of communication on the Web. Despite the recent increase in usage of social networks, message continues to be the primary medium for billions of users across the world to connect and share information [2]. With the rapid increase in message overload, it has become increasingly challenging for users to process and respond to incoming messages. It can be especially time-consuming to type message replies on a wearable device.

In Computer Science, the field of AI research defines itself as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of success at some goal. The term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving". As machines become increasingly capable, mental facilities once thought to require intelligence are removed from the definition.

Wearable and the Internet of Things (IoT) may give the impression that it's all about the sensors, hardware, communication middleware, network and data but the real value is in insights. In this article, we explore artificial intelligence (AI) and machine learning that are becoming indispensable tools for insights.

Artificial Intelligence: The field of artificial intelligence is the study and design of intelligent agents able to perform tasks that require human intelligence, such as visual perception, speech recognition, and decision-making. In order to pass the Turing test, intelligence must be able to reason, represent knowledge, plan, learn, communicate in natural language and integrate all these skills towards a common goal.

Machine Learning: The machine learning is the subfield that learns and adapts automatically through experience. It focuses on prediction, based on properties learned from the training data. The origin of machine learning can be traced back to the development of neural network model and later to the decision tree method. Supervised and unsupervised learning algorithms are used to predict the outcome based on the data.

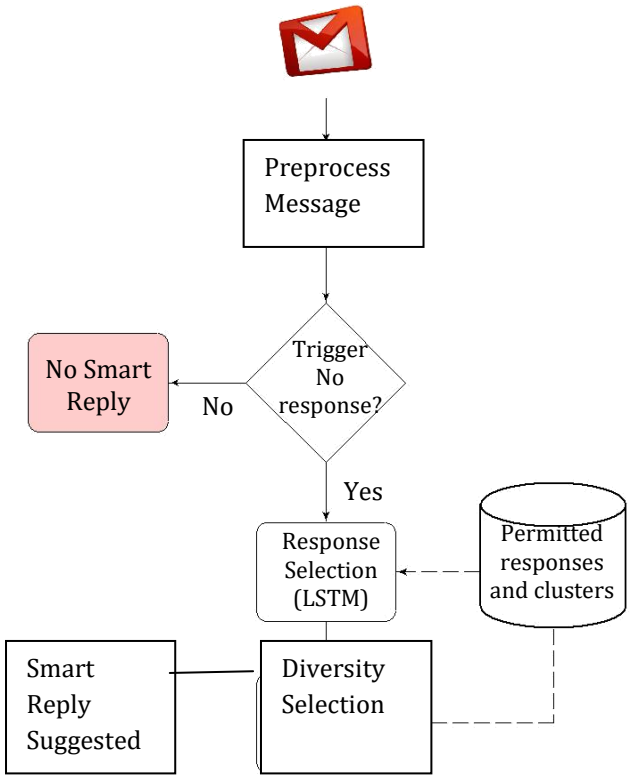

Fig -1 The Components of Smart Reply

To address this problem, we leverage the sequence-to-sequence learning framework [23], which uses long short-term memory networks (LSTMs) [10] to predict sequences of text. Consistent with the approach of the Neural Conversation Model [24], our input sequence is an incoming message and our output distribution is over the space of possible replies. Training this framework on a large corpus of conversation data produces a fully generative model that can produce a response to any sequence of input text. As Neural Conversion model [24] can be used to decode coherent, possible responses.

We have faced several challenges not considered in previous work:

- Response quality
- Utility
- Scalability
- Privacy

To tackle these challenges, we propose Smart Reply a novel method and system for automated message response suggestion. Smart Reply Consists of the following components, also shown in Fig-1.

- **Response selection:** At the core of our system, an LSTM neural network [8] processes an incoming message, and then uses it to predict the most likely responses. LSTM computation can be expensive, so we improve scalability by ending only the approximate best responses.
- **Response set generation:** To deliver high response quality, we only select responses from response space which is generated in using a semi-supervised graph learning approach.
- **Diversity:** After ending a set of most likely responses from the LSTM, we would like to choose a small set to show to the users that maximize the total utility. We found that enforcing diverse semantic intents is critical to making the suggestions useful.
- **Triggering model:** The triggering model is the entry point of Smart Reply system. It is responsible for iterating messages that are bad candidates for suggesting responses.

The combination of these components is a novel end-to-end method for generating short, complete responses to messages, going beyond previous works. For response selection it exploits state-of-the-art deep learning models trained on billions of words, and for response set generation it introduces a new semi-supervised method for semantic understanding of user-generated content.

Moreover, since it directly addresses all four challenges mentioned above, it has been successfully deployed in Inbox. Currently, the Smart Reply system is responsible for assisting with 10% of message replies for Inbox on wearable.

2. RELATED WORK

As we consider related work, we note that building an automated system to suggest message responses is not a task for an existing literature or benchmarks, this is a standard machine learning problem to which existing algorithms can be readily applied. The work related to two of our core components which we will review here.

Predicting full responses: Full response prediction was initially attempted in Social media [16], which approached the problem from the perspective of machine translation. Our approach is similar, but rather than using SMT, we use the neural network translation model proposed in "sequence-to-sequence learning".

Sequence-to-sequence learning: which makes use of long short-term memory networks (LSTMs) [10] to predict sequences of text, was originally applied to Machine Translation but has since seen success in other domains such as image caption[25] and speech recognition [6]

Other recent works have also applied recurrent neural net-works (RNNs) or LSTMs to full response prediction [21], [20], [19], [24]. In [21] the authors rely on having an SMT system to generate n-best lists, while [19] and [24], like this work, develop fully generative models. Our approach is most similar to the Neural Conversation Model [24], which uses sequence-to-sequence learning to model tech support chats and movie subtitles.

The primary difference of our work is that it was deployed in a production setting, which raised the challenges of response quality, utility, scalability, and privacy. These challenges were not considered in any of these related works and led to our novel solutions explained in the rest of this paper.

Furthermore, in this work we approach a different domain than [21], [20], [19], and [24], which primarily focus on social media and movie dialogues. In both of those domains it can be acceptable to provide a response that is merely related or on-topic. Message, on the other hand, frequently expresses a request or intent which must be addressed in the target response space. Our approach here builds on the Expander graph learning approach [15], since it scales well to both large data and large output sizes. While Expander was originally proposed for knowledge expansion and classification tasks [26], our work is the rest to use it to discover semantic intent clusters from user-generated content.

Other graph-based semi-supervised learning techniques have been explored in the past for more traditional classification problems [27][5]. Other related works have explored tasks involving semantic classification [12] or identifying word-level intents [17] targeted towards Web search queries and other forums [7]. However, the problem

settings and tasks them-selves are significantly different from what is addressed in our work.

Finally, we note that Smart Reply is the rest work to address these tasks together and solve them in a single end-to-end, deployable system.

3. SELECTING RESPONSES

The fundamental task of the Smart Reply system is to find the most likely response given an original message. In other words, given original message o and the set of all possible responses R, we would like to find:

$$r = \text{argmax } P(r|o)$$

$$r \in R$$

To find this response, we will construct a model that can score responses and then find the highest scoring response. We will next describe how the model is formulated, trained, and used for inference. Then we will discuss the core challenges of bringing this model to produce high quality suggestions on a large scale.

3.1 LSTM Model

Since we are scoring one sequence of tokens r, conditional on another sequence of tokens o, this problem is a natural t for sequence-to-sequence learning [23]. The model itself is an LSTM. The input is the tokens of the original message P(rjo) ::: o_ng, and the output is the conditional probability distribution of the sequence of response tokens given the input:

$$P(r_1; :::; r_m|o_1; :::; o_n)$$

As in sequence-to-sequence learning [23], this distribution can be factorized as:

$$P(r_1; :::; r_m|o_1; :::; o_n) = \prod_{i=1}^{y^m} P(r_i|o_1; :::; o_n; r_1; :::; r_{i-1})$$

First, the sequence of original message tokens, including a special end-of-message token o_n, are read in, such that the LSTM's hidden state encodes a vector representation of the whole message. Then, given this hidden state, a softmax output is computed and interpreted as $P(r_1|o_1; :::; o_n)$, or the probability distribution for the rest response token. As response tokens are fed in, the softmax at each time step t is interpreted as $P(r_t|o_1; :::; o_n; r_1; :::; r_{t-1})$. Given the factorization above, these softmax can be used to compute $P(r_1; :::; r_m|o_1; :::; o_n)$.

Training Given a large corpus of messages, the training objective is to maximize the log probability of observed responses, given their respective originals:

$$\sum_{(o;r)} \log P(r1; :::; rmjo1; :::; on)$$

We train against this objective using stochastic gradient descent with AdaGrad [8]. Ten epochs are run over a message corpus which will be described in Section 7.1. Due to the size of the corpus, training is run in a distributed fashion using the TensorFlow library [1].

Both our input and output vocabularies consist of the most frequent English words in our training data after pre-processing steps.. In addition to the standard LSTM formulation, we found that the addition of a recurrent projection layer [18] substantially improved both the quality of the converged model and the time to converge. We also found that gradient clipping was essential to stable training. These distributions can be used in a variety of ways.

- To draw a random sample from the response distribution $P(r_1; :::; r_m|o_1; :::; o_n)$. This can be done by sampling one token at each time step and feeding it back into the model.
- To determine the most likely response, given the original message. This can be done greedily by taking the most likely token at each time step and feeding it back in. A less greedy strategy is to use a beam search, i.e., take the top b tokens and feed them in, then retain the b best response softmax and repeat the procedure.
- To determine the likelihood of a specific response candidate. This can be done by feeding in each token of the candidate and using the softmax output to get the likelihood of the next candidate token.

Table - 1 Generated response examples.

Query	Top generated responses
Hi, I thought it would be great for us to sit down and chat. I am free Tuesday and Wednesday.	I can do Tuesday. I can do Wednesday. How about Tuesday? I can do Tuesday!
Can you do either of those days?	I can do Tuesday. What time works for you? I can do Wednesday!
Thanks!	I can do Tuesday or Wednesday.
{Alice	How about Wednesday? I can do Wednesday. What time works for you? I can do either.

3.2 Challenges

As described thus far, the model can generate coherent and plausible responses given an incoming message. However, several key challenges arise when bringing this model into production.

Given that the model is trained on a corpus of real messages, we have to account for the possibility that the most probable response is not necessarily a high quality response. Even a response that occurs frequently in our corpus may not be appropriate to surface back to users.

While restricting the model vocabulary might address simple cases such as profanity and spelling errors, it would not be sufficient to capture the wide variability.

We further improve the utility of suggestions by rest passing each message through a triggering model that determines whether suggestions should be generated at all. This reduces the likelihood that we show suggestions when they would not be used anyway.

Scalability Our model needs to be deployed in a production setting and cannot introduce latency to the process of message delivery, so scalability is critical.

Our search is conducted as follows. First, the elements of R are organized into a tree. Then, we conduct a left-to-right beam search, but only retain hypotheses that appear in the tree. This search process has complexity O(bl) for beam size b and maximum response length l. Both b and l are typically in the range of 10-30, so this method dramatically reduces the time to find the top responses and is a critical element of making this system deployable. In terms of quality, we find that, although this search only approximates the best responses in R, its results are very similar to what we would get by scoring and ranking all r 2 R, even for small b. At b = 128, for example, the top scoring response found by this process matches the true top scoring response 99% of the time. Results for various beam sizes are shown in Fig- 2.

Privacy Note that all message data was encrypted. Engineers could only inspect aggregated statistics on anonym zed sentences that occurred across many users and did not identify any user. Also, only frequent words are retained. As a result, verifying model's quality and debugging is more complex.

For a sample of messages we compute the frequency which the best candidate found by a beam search over R matches the best candidate found by exhaustively scoring all members of R. We compare various beam sizes. At a beam size of 16, these two methods find the same best response 93% of the time.

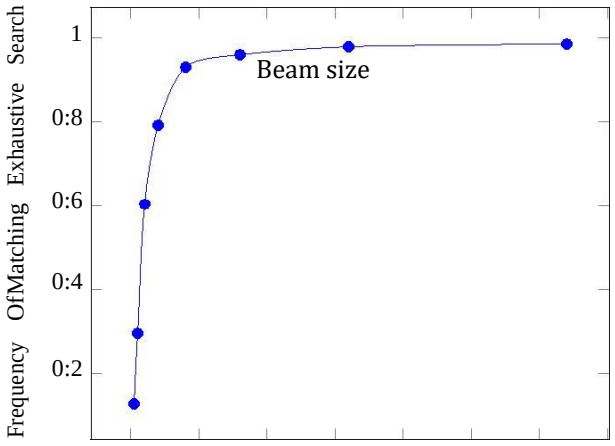

Fig -2 Effectiveness of searching the response space R

4. RESPONSE SET GENERATION

Two of the core challenges we face when building the end to end automated response system are response quality and utility. Response quality comes from suggesting “high quality" responses that deliver a positive user experience. Utility comes from ensuring that we don't suggest multiple responses that capture the same intent. We can consider these two challenges jointly.

We first need to be a target response space that comprises high quality messages which can be surfaced as suggestions. The goal here is to generate a structured response set that effectively captures various intents conveyed by people in natural language conversations. The target response space should capture both variability in language and intents. The result is used in two ways downstream (a) To be a response space for scoring and selecting suggestions using the model described in Section 3, and (b) promote diversity among chosen suggestions as discussed in Section 5.

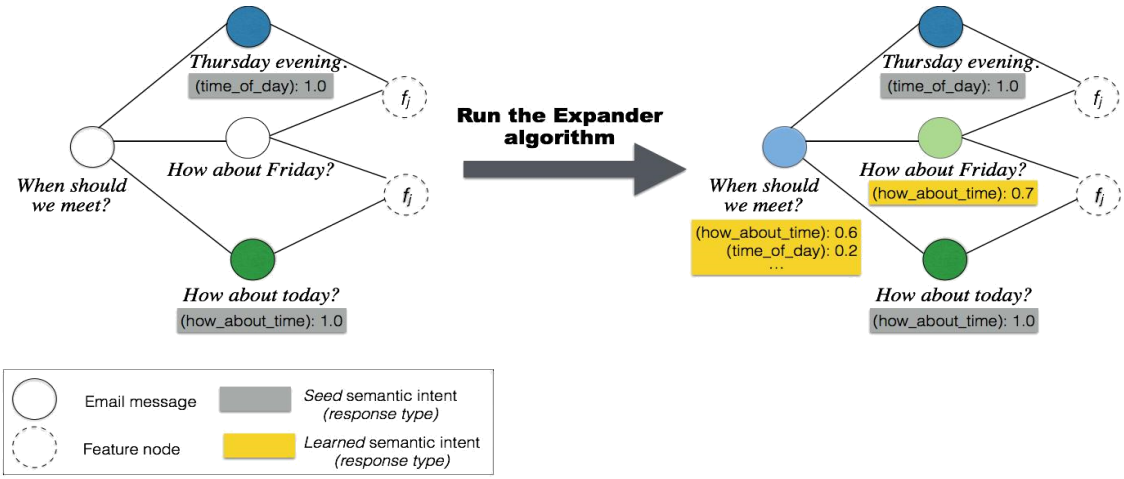

Fig-3 Semantic clustering of response messages

We construct a response set using only the most frequent anonymized sentences aggregated from the preprocessed data.

4.1 Canonicalizing message responses

The First step is to automatically generate a set of canonical responses messages that capture the variability in language. We parse each sentence using a dependency parser and use its syntactic structure to generate a canonicalized representation. Words that are modifiers or unattached to head words are ignored.

4.2 Semantic intent clustering

In the next step, we want to partition all response messages into “semantic" clusters where a cluster represents meaningful response intent. All messages within a cluster share the same semantic meaning but may appear very different.

In this step helps to automatically digest the entire information present in frequent responses into a coherent set of semantic clusters. If we were to build a semantic intent prediction model for this purpose, we would need access to a large corpus of sentences annotated with their corresponding semantic intents. However, this is neither readily available for our task nor at this scale. Moreover, unlike typical machine learning classification tasks, the semantic intent space cannot fully need a priori as shown in Figure 3.

4.3 Graph construction

We start with a few manually clusters sampled from the top frequent messages. A small number of example responses are added as “seeds" for each cluster. We then construct a base graph with frequent response messages as nodes (V_R). For each response message, we further extract a set of lexical features and add these as “feature" nodes (V_F) to the same graph. Edges are created between a pair of nodes (u, v) where u 2 V_R and v 2 V_F if v belongs to the feature set for response u. We follow the same process and create nodes for the manually labeled examples V_L. We make an observation that in some cases an incoming original message could potentially be treated as a response to another message depending on the context.

4.4 Semi supervised learning

The constructed graph captures relationships between similar canonicalized responses via the feature nodes. Next, we learn a semantic labeling for all response nodes by propagating semantic intent information from the manually labeled examples through the graph.

We treat this as a semi-supervised learning problem and use the distributed Expander [15] framework for optimization. The learning framework is scalable and naturally suited for semi-supervised graph propagation tasks such as the semantic clustering problem described here. We minimize the following objective function for response nodes in the graph.

The output from Expander is a learned distribution of semantic labels for every node in the graph. We assign the top scoring output label as the semantic intent for the node; labels with low scores are iterated out.

To discover new clusters which are not covered by the labeled examples, we run the semi-supervised learning algorithm in repeated phases. In the First phase, we run the label propagation algorithm for 5 iterations. We then x the cluster assignment, randomly sample 100 new responses from the remaining unlabeled nodes in the graph. The sampled nodes are treated as potential new clusters and labeled with their canonicalized representation. We return label propagation with the new labeled set of clusters and repeat this procedure until convergence.

5. SUGGESTION DIVERSITY

The LSTM processes an incoming message and then selects the best responses from the target response set

created using the method described in Section 4. Recall that we follow this by some light normalization to positive responses that may be too general to be valuable to the user. The effect of this normalization can be seen by comparing columns 1 and 2 of Table 2.

Next, we need to choose a small number of options to show the user. A straight-forward approach would be to just choose the N top responses and present them to the user. However, as column 2 of Table 2 shows, these responses tend to be very similar.

The likelihood of at least one response being useful is greatest when the response options are not redundant, so it would be wasteful to present the user with three variations of, say, I'll be there. The job of the diversity component is to select a more varied set of suggestions using two strategies: omitting redundant responses and enforcing negative or positive responses.

5.1 Omitting Redundant Responses

This strategy assumes that the user should never see two responses of the same intent. It can be thought of as a cluster of responses that have a common communication purpose. In Smart reply, every target response is associated in exactly one intents are based on automatic clustering followed by human validation as discussed in Section 4.

The actual diversity strategy is simple: the top responses are iterated over in the order of decreasing score. Each response is added to the list of suggestions, unless its intent is already covered by a response on the suggestion list. The resulting list contains only the highest scored representative of each intent, and these representatives are ordered by decreasing score.

5.2 Enforcing Negatives and Positives

We have observed that the LSTM has a strong tendency towards producing positive responses, whereas negative responses typically receive low scores. We believe that this tendency affects the style of message conversations: positive replies may be more common, and where negative responses are appropriate, users may prefer a less direct wording.

Nevertheless, we think that it is important to or negative suggestions in order to give the user a real choice.

A positive response is one which is clearly amative. In order to the negative response, a second LSTM pass is performed. In this second pass, the search is restricted to only the negative responses in the target set. This is necessary since the top responses produced in the pass may not contain any negatives.

Even though missing negatives are more common, there are some cases in which an incoming message triggers exclusively negative responses. In this situation, we employ an analogous strategy for enforcing a positive response. The Final set of top scoring responses is then presented to the user as suggestions.

6. TRIGGERING

The triggering module is the entry point of the Smart Reply system. It is responsible for altering messages that are bad candidates for suggesting responses. This includes messages for which short replies are not appropriate as well as messages for which no reply is necessary at all.

The module is applied to every incoming message just after the preprocessing step. If the decision is negative, we wish the execution and do not show any suggestions. Currently, the system decides to produce a Smart Reply for roughly 11% of messages, so this process vastly reduces the number of useless suggestions seen by the users. An additional benefit is to decrease the number of calls to the more expensive LSTM inference, which translates into smaller infrastructure cost.

There are two main requirements for the design of the triggering component. First, it has to be good enough to grow out cases where the response is not expected. Note that this is a very different goal than just scoring a set of responses. For instance, we could propose several valid replies to a newsletter containing a sentence "Where do you want to go today?". Second, it has to be fast: it processes hundreds of millions of messages daily, so we aim to process each message within milliseconds.

The main part of the triggering component is a feed forward neural network which produces a probability score for every incoming message. If the score is above some threshold, we trigger and run the LSTM scoring. We have adopted this approach because feed forward networks have repeatedly shown to perform linear models such as SVM or linear regression on various NLP tasks.

Table -2 Different response rankings for the message Can you join tomorrow's meeting?"

Unnormalized Responses	Normalized Responses
Yes, I'll be there.	Sure, I'll be there.
Yes, I will be there.	Yes, I can.
I'll be there.	Yes, I can be there.
Yes, I can.	Yes, I'll be there.
What time?	Sure, I can be there.
I'll be there!	Yeah, I can.
I will be there.	Yeah, I'll be there.
Sure, I'll be there.	Sure, I can.
Yes, I can be there.	Yes. I can.
Yes!	Yes, I will be there.

Normalized Negative Responses
Sorry, I won't be able to make it tomorrow. Unfortunately I can't. Sorry, I won't be able to join you. Sorry, I can't make it tomorrow. No, I can't. Sorry, I won't be able to make it today. Sorry, I can't. I will not be available tomorrow. I won't be available tomorrow. Unfortunately, I can't.
Final Suggestions
Sure, I'll be there. Yes, I can. Sorry, I won't be able to make it tomorrow.

6.1 Data and Features

The data set described in Section 7.1, we create a training set that consists of pairs (o, y), where o is an incoming message and y is true or false is a Boolean label, which is true if the message had a response and false otherwise. For the positive class, we consider only messages that were replied to from a wearable device, while for negative we use a subset of all messages. Our goal is to model P (y = true j o), the probability that message o will have a response on wearable.

After preprocessing, we extract content features from the message body, subject and headers. We also use various social signals like whether the sender is in recipient's address book, whether the sender is in recipient's social network and whether the recipient responded in the past to this sender.

6.2 Network Architecture and Training

We use a feed forward multilayer perception with an embedding layer and three fully connected hidden layers. We use feature hashing to bucket rare words that are not present in the vocabulary. The embeddings are separate for each sparse feature type. Then, all sparse feature embeddings are concatenated with each other and with the vector of defense features.

We use Rectified [13] activation function for non-linearity between layers, the dropout [22] layer is applied after hidden layer. We train the model using large scale machine learning adaptive sub gradient [8] optimization algorithm with logistic loss cost function.

7. EVALUATION AND RESULTS

In this section, we describe the training and test data for all messages using various preprocessing steps. Then, we evaluate different components of the Smart Reply system and present overall usage statistics

7.1 Data

To generate the training data for all Smart Reply models from sampled accounts, we extracted all pairs of an incoming message and the user's response to that message. For training the triggering model, all sampled number of incoming personal messages which the user didn't reply to. At the beginning of Smart Reply pipeline data is preprocessed in the following way:

- **Language detection:** The language of the message is identified and non-English messages are discarded.
- **Tokenization:** Subject and message body are broken into words and punctuation marks.
- **Sentence segmentation:** Sentences boundaries are identified in the message body.
- **Normalization:** Infrequent words and entities like personal names, URLs, message addresses, phone numbers etc. are replaced by special tokens.
- **Quotation removal:** Quoted original messages and forwarded messages are removed.
- **Salutation/Close removal:** Salutations like Hi John and closes such as Best regards, Mary are removed.

After the preprocessing steps, the size of the training set is 238 million messages, which include 153 million messages that have no response.

7.2 Results

The most important end-to-end metric for our system is the fraction of messages for which it was used. This is currently 10% of all wearable replies. Below we describe in more detail evaluation starts for different components of the system.

7.2.1 Triggering results

In order to evaluate the triggering model, we split the data set into train (80%) and test (20%) such that all test messages are delivered after train messages. This is to ensure that the test conditions are similar to the Final scenario. We use a set of standard binary classifier metrics: precision, recall and the area under the ROC curve. The AUC of the triggering model is 0:854. We also compute the fraction of triggered messages in the deployed system, which is 11%. We observed that it may be beneficial to slightly over-trigger, since the cost of presenting a suggestion, even if it is not used, is quite low

7.2.2 Response selection results Perplexity.

A model with lower perplexity assigns higher likelihood to the test responses, so we expect it to be better at predicting responses. Intuitively, a perplexity equal to k means that when the model predicts the next

word, there are on average k likely candidates. In particular, for the ideal scenario of perplexity equal to 1, we always know exactly what should be the next word. The perplexity on a set of N test samples is computed using the following formula:

$$P_r = \exp\left(\frac{1}{W}\sum_{X_i=1}^{N} \ln(P(r_1; :::; r_m | o_1 ; :::; o_n))\right)$$

^

Where W is the total number of words in all N samples, P is the learned distribution and r^i, o^i are the i-th response and original message. Note that in the equation above only response terms are factored into P_r. The perplexity of the Smart Reply LSTM is 17:0. By comparison, an n-grams language model with S.M Katz et.al [11] and a maximum order of 5 has a perplexity of 31:4 on the same data response ranking.

While perplexity is a quality indicator, it does not actually measure performance at the scoring task we are ultimately interested in. In particular, it does not take into account the constraint of choosing a response in R. Therefore we also evaluate the model on a response ranking task: for each of N test message pairs (o; r) for which r 2 R, we compute s = P (rjo) and 8_i x_i = P ($w_i$jo), where w_i is the i-th element of R. Then we sort the set R = fs; x_1; : : : ; x_N g in descending order. Finally, we de ne $rank_i$ = $argmin_j$(R_jjR_j = s). Put simply, we are ending the rank of the actual response with respect to all elements in R

Table -3 Response ranking

Model	Precision@10	Precision@20	MRR
Random	5:58e 4	1:12e 3	3:64e 4
Frequency	0:321	0:368	0:155
Multiclass-BOW	0:345	0:425	0:197
Smart Reply	0:483	0:579	0:267

Using this value, we can compute the Mean Reciprocal Rank:

$$MRR = 1/N \sum_{X_i}^{N} 1/rank_i$$

Additionally we can compute, for a given value of K it is computed as the number of cases for which target response r was within the top K responses that were ranked by the model.

We compare the Smart Reply response selection model to three baselines on the same ranking task. The Random baseline ranks R randomly. The Frequency baseline ranks them in order of their frequency in the training corpus. This baseline captures the extent to which we can simply suggest highly frequent responses without regard for the contents of the original message. The Multiclass-BOW baseline ranks R using a feed forward neural network whose input is the original message, represented with bag of words features, and whose output is a distribution over the elements of R(a softmax).

As shown in Table 3, the Smart Reply LSTM significantly improves on the Frequency baseline, demonstrating that conditioning on the original message is effective; the model successfully extracts information from the original message and uses it to rank responses more accurately.

It also significantly outperforms the Multiclass-BOW base line. There are a few possible explanations for this. First, the recurrent architecture allows the model to learn more sophisticated language understanding than bag of words features. Second, when we pose this as a multiclass prediction problem, we can only train on messages whose response is in R, a small fraction of our data. On the other hand, the sequence-to-sequence framework allows us to take advantage of all data in our corpus: the model can learn a lot about original-response relationships even when the response does not appear in R exactly.

Note that an added disadvantage of the multiclass formulation is that it tightly couples the training of the model to the construction of R. We expect R to grow over time, given the incredible diversity with which people communicate. While a simpler application such as chat might only need a small number of possible responses, we nd that for message we will need a tremendous number of possible suggestions to really address users' needs.

7.2.3 Diversity results

We justify the need for both the diversity component and a sizable response space R by reporting statistics around unique suggestions and clusters in Table 4. The Smart Re-ply system generates daily 12:9k unique suggestions that belong to 376 unique semantic clusters. Out of those, people decide to use 4; 115, or 31:9% of, unique suggestions and 313, or 83:2% of, unique clusters. Note, however, that many suggestions are never seen, as column 2 shows: the user may not open a message, use the web interface instead of wearable or just not scroll down to the bottom of the message. Also, only one of the three displayed suggestions will be selected by the user. These statistics demonstrate the need to go well beyond a simple system with 5 or 10 canned responses.

Table -4 Unique cluster/suggestions usage per day

	Daily Count	Seen	Used
Unique Clusters	376	97:1%	83:2%
Unique Suggestions	12:9k	78%	31:9%

The distribution of the rank for suggested responses and the distribution of suggested clusters. The tail of the cluster distribution is long, which explains the poor performance of Frequency baseline described in Section 7.2.2.

We also measured how Smart Reply suggestions are used based on their location on a screen. Recall that Smart Reply always presents 3 suggestions, where they are suggestion is the top one. We observed that, out of all used suggestions, 45% were from the 1st position, 35% from the 2nd position and 20% from the 3rd position. Since usually the third position is used for diverse responses, we conclude that the diversity component is crucial for the system quality.

Finally, we measured the impact of enforcing a diverse set of responses (e.g., by not showing two responses from the same semantic cluster) on user engagement: when we completely disabled the diversity component and simply suggested the three suggestions with the highest scores, the click-through rate decreased by roughly 7:5% relative.

8. CONCLUSIONS

We presented Smart Reply, a novel end-to-end system for automatically generating short, complete message responses. The core of the system is a state-of-the-art deep LSTM model that can predict full responses, given an incoming message. To successfully deploy this system in Inbox by Message, we addressed several challenges:

We increase the total utility of our chosen combination of suggestions by enforcing diversity among them, and altering track for which suggestions would not be useful.

Our clearest metric of success is the fact that 10% of mo-bile replies in Inbox are now composed with assistance from the Smart Reply system. Furthermore, we have designed the system in such a way that it is easily extendable to address additional user needs; for instance, the architecture of our core response scoring model is language agnostic, therefore accommodates extension to other languages in addition to English

9. REFERENCES

[1] Machine learning on heterogeneous systems. 2015.

[2] I. G. P. A airs. Interconnected world: Communication & social networking. Press Release, March 2012. http://www.ipsos-na.com/news-polls/ pressrelease.aspx?id=5564.

[3] Y. Artzi, P. Pantel, and M. Gamon. Predicting responses to microblog posts. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 602-606, Montreal, Canada, June 2012. Association for Computational Linguistics.

[4] L. Backstrom, J. Kleinberg, L. Lee, and C.Danescu-Niculescu-Mizil. Characterizing and curating conversation threads: Expansion, focus, volume, re-entry. In Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, WSDM '13, pages 13{22, 2013.

[5] Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. In O. Chapelle, B. Scholkopf, and Zien, editors, Semi-Supervised Learning, pages 193, 216. MIT Press, 2006.

[6] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals. Listen, attend, and spell. arXiv:1508.01211, abs/1508.01211, 2015.

[7] Z. Chen, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh. Identifying intention posts in discussion forums. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1041-1050, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[8] J. Duchi, E. Hazad, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. JMLR, 12, 2011.

[9] Y. Goldberg. A primer on neural network models for natural language processing. CoRR, abs/1510.00726, 2015.

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735{1780, 1997.

[11] S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recogniser. IEEE Transactions on Acoustics, Speech, and Signal Processing, 35:pages 400-401, 1987.

[12] X. Li. Understanding the semantic structure of noun phrase queries. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL), pages 1337-1345, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[13] V. Nair and G. E. Hinton. Recti ed linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), pages 807-814, 2010.

[14] B. Pang and S. Ravi. Revisiting the predictability of language: Response completion in social media. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and

Computational Natural Language Learning, pages 1489-1499, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[15] S. Ravi and Q. Diao. Large scale distributed semi-supervised learning using streaming approximation. In Proceedings of the 19th International Conference on Arti cial Intelligence and Statistics (AISTATS), 2016.

[16] A. Ritter, C. Cherry, and W. B. Dolan. Data-driven response generation in social media. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, Edinburgh, UK, July 2011. Association for Computational Linguistics.

[17] R. Saha Roy, R. Katare, N. Ganguly, S. Laxman, and M.Choudhury. Discovering and understanding word level user intent in web search queries. Web Semant., 30(C):22{38, Jan. 2015.

[18] H. Sak, A. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH), 2014.

[19] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J.Pineau. Hierarchical neural network generative models for movie dialogues. In arXiv preprint arXiv:1507.04808, 2015.

[20] L. Shang, Z. Lu, and H. Li. Neural responding machine for short-text conversation. In In Proceedings of ACL-IJCNLP, 2015.

[21] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversation responses. In In Proceedings of NAACL-HLT, 2015.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R.Salakhutdinov. Dropout: A simple way to prevent neural networks from over tting. Journal of Machine Learning Research, 15:pages 1929-1958, 2014.

[23] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems (NIPS), 2014.

[24] O. Vinyals and Q. V. Le. A neural conversation model. In ICML Deep Learning Workshop, 2015.

[25] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[26] J. B. Wendt, M. Bendersky, L. Garcia-Pueyo, V. Josifovski B.Miklos, I. Krka, A. Saikia, J. Yang, M.-A. Cartright, and S. Ravi. Hierarchical label propagation and discovery for machine generated message. In Proceedings of the International Conference on Web Search and Data Mining (WSDM) (2016), 2016.

[27] X. Zhu, Z. Ghahramani, and J. La erty. Semi-supervised learning using gaussian elds and harmonic functions. In Proceedings of the International Conference on Machine Learning.