Neural Models for Predicting Email Response Behavior
Stanford CS224U - Spring 2018

Tucker Leavitt
Stanford University
tuckerl@cs.stanford.edu

Abstract

Modern email applications often highlight important or anomalous emails to users. Towards improving these applications, we develop a neural network for the task of predicting whether a user will respond to an email, given the text of the email. We derive an email response dataset from the Enron email corpus. Using this dataset, we train two types of models: a bi-directional LSTM with attention, and a collaborative filtering model that incorporates sender and recipient information. Our best model achieves an F1 score of 77.02, outperforming our baselines and existing machine learning frameworks for the task.

1 Introduction

Despite its age, email remains one of the most popular online activities. Email is the primary communication channel for talking across organizational boundaries, though in many ways it is outdated. Products like Google Inbox are starting to bring intelligent recommendations and better organization to email. The advances in natural language processing over the past four years have great potential for applications that improve email.

Towards creating a more intelligent email experience, this project seeks to create a neural network for the task of **predicting whether the recipient of an email will reply or forward it**. This reduces the general problem of recommending emails to a tractable binary classification task. Machine learning techniques have been applied to the email reply prediction task in the past, but there has been little work on this task that leverages recent advances in deep learning and word representations.

In this paper, we review published literature on email reply prediction and related problems in section 2. We then discuss the approach we take for the email reply prediction problem in section 3. We overview the dataset we created to train our models in section 4. We describe our models' configuration in section 5.1 and summarize our models' performance in section 5.2. The results are discussed in section 6 and future work is proposed in section 7.

2 Previous Work

Several works have tackled the problem of email response prediction over the past decade. Earlier works used hand-built feature extractors and linear models, and more recent approaches have used neural networks.

In 2009, researchers at the University of Portsmouth used manually defined features and a rudimentary scoring algorithm to categorize emails as 'need a response' and 'no response needed' (Taiwo Ayodele and Khusainov, 2009). They assigned each email a score and predict 'need a response' if the score is above a certain threshold. They update the score by applying if-then decision rules like "if the subject contains a question, increment the score by 3." They achieve high accuracy compared to a "gold standard" of human-classified emails, but their dataset is very small (only 5000 emails), so their results are likely to scale poorly. A large bottleneck in the field of email reply prediction seems to be access to labeled data.

In 2017, researchers from Microsoft and UMass Amherst (Yang et al., 2017) used a logistic regression classifier trained on curated domain-specific features to predict whether a recipient

will reply to an email and how long it will take to do so. Their features included a bag-of-words text model, individual and pair-wise historical interactions, the time of day at which the email was sent, and the job titles of the senders and recipients, among other features. They report an AUROC of 0.72 for reply prediction and a classification accuracy of 42% for predicting response time. They did not share features between their models; it seems like information about when or if an email will be responded to might be useful for doing reply prediction. I was impressed (and a bit indimidated) by the sophistication and breadth of the features they engineered. I hope it is not necessary to do such feature engineering to get good performance, though in the case that feature engineering is the best way to go, this paper will be an excellent resource.

In 2015, two Stanford students in CS224D trained an LSTM and a Convolutional Neural Network (CNN) using unigram and bigram bag-of-words model to predict "intent" for around 10,000 sentences from the Enron email corpus (i.e. whether the sentences solicited a response) (Eugene and Caswell, 2015) . Their best model, the CNN, achieves an F1 score of 0.81, though their loss curves indicate overfitting, and their Random Forest Baseline outperforms their deep learning models on some variations of the main task. Their results indicate that complex feature engineering like that in (Yang et al., 2017) is not necessary for achieving good performance on this task.

The Enron email corpus has been annotated with the corporate role (CEO, Manager, Employee, etc.) of many of the email authors and recipients. In 2011, Bramsen et al. (Bramsen et al., 2011) used this annotated dataset to build a model for predicting whether emails where "UpSpeak" (from a subordinate to a superior), "DownSpeak" (from a superior to a subordinate), or "PeerSpeak" (between peers). They tried several feature extraction techniques and classifier models: their best features were word n-grams which they grouped into bins by frequency, and extractors that look for "polite imperatives" (imperatives pre- ceded by the word "please"), and their best model was a SVM. This model achieved a F1-score of 79.7 on a binary UpSpeak/DownSpeak email classification task. When it was published, this paper was the first work to present a machine learning model for Social Power Modeling with email. The idea of modeling peer relationships, as in this work, may also be useful for tasks such as email reply prediction.

The author was unable to find prior work on email reply prediction that incorporated information about the releationship of the sender and recipient. This may be because most techniques for leveraging user relationships to predict item interaction probabilities rely on factorizing the user-item interaction matrix. Constructing such a matrix assumes a describable set of items, and it is not clear how to describe the set of all emails in a way that leads to a sensible user-item interaction matrix. However, in 2017, a team of researchers from Singapore and Columbia University described (He et al., 2017) a 'neural' collaborative filtering technique that can be applied in more general settings. The idea is to learn user embeddings, item embeddings, and an interaction function via backpropagation on a network for predicting the user-item interaction probability. They found that their methods outperformed their baselines and state-of-the art matrix factorization techniques on the MovieLens dataset. Unfortunately, their methods do not extend directly to the email reply prediction problem, since a single email involves several users.

3 Methods

Previous approaches (Eugene and Caswell, 2015) (Yang et al., 2017) have demonstrated that email word vectors can be effective input features for email response models. We use this as the basis of our model, but also incorporate information about the relationship between senders and recipients.

In this project, we implement and evaluate the following models:

1. a random baseline classifier (i.e. model that generates class predictions uniformly at random), referred to as Random Classifier,

2. a linear logistic regression classifier, referred to as Linear LogReg,

3. a bi-directional LSTM with attention weighting, referred to as Bi-RNN, and

4. a bi-directional LSTM (identical to the previous model) with an added sender-recipient interaction term, referred to as Bi-RNN w/ SRI

3.1 Sequence Model Architecture

We use a variant of the Heirarchical Attention Network (Yang et al., 2016), or HAN, to learn document vectors for the emails. The HAN architecture is composed of two bi-directional RNNs: one for learning sentence embeddings from word vectors, and another for learning document embeddings from sentence embeddings. Yang et al. use a self-attention mechanism at both the sentence and document level, which enables the model to focus on important content when constructing a document representation. To due the brevity of many emails, we implemented a single-level hierarchy with only one RNN. This amounts to treating an email as a single sentence, using the one sentence vector as the document representation.

Figure 1 shows a schematic diagram of the model. We use GloVe vectors (Pennington et al., 2014) as our word representations. The attention layer computes a weighted sum s of the LSTM hidden states h_t as follows:

$$u_t = \tanh(W_w h_t + b_w)$$
$$\alpha_t = \frac{\exp(u_t^T u)}{\sum_t \exp(u_t^T u)}$$
$$s = \sum_t \alpha_t h_t$$

where W_w, b_w, and u are model parameters.

3.2 Incorporating Sender-Recipient Relationships

To incorporate information about the sender and recipients, construct a variant of the Neural Matrix Factorization model proposed in (He et al., 2017). This model attempts to predict user-item interactions by representing users and items in a latent vector space (called the embedding space), and training a two-pronged neural network on the user and item vectors. The authors report state-of-the-art results on the MovieLens and Pinterest collaborative filtering datasets with this model (He et al., 2017).

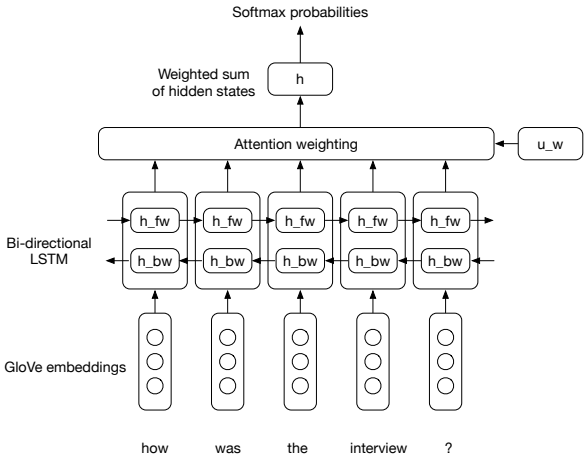

Figure 1: Schematic diagram of the bi-directional LSTM model.

The email response prediction problem is slightly more complicated than MovieLens and Pinterest, since it cannot be modeled as a single user-item interaction. Each email involves three parties: the sender, the recipient, and the email itself. The goal of the model is to predict whether the receiver will interact with the email.

One way to adapt this problem to the framework of (He et al., 2017) is to combine the sender and email vectors into a single 'item' representation of the email. To this end, we propose the following model for the probability $\hat{y}_{abu}$ that user a interacts with (i.e. responds to) an email u sent by another user b:

$$\phi_{user} = p_a \odot \left(W_u \begin{bmatrix} p_b \\ q_u \end{bmatrix} + b_u \right) \quad (1)$$

$$\phi_{item} = q_u \quad (2)$$

$$\hat{y}_{abu} = \sigma \left(h^T \begin{bmatrix} \phi_{user} \\ \phi_{item} \end{bmatrix} \right) \quad (3)$$

Here, p_a and p_b are embedding vectors for the recipient and sender, and q_u is the document vector representation of the email. This can in principle be any document representation; here we use the output of the bi-rnn model (see section 3.1). W_u, b_u, and h are model parameters, and σ is the sigmoid function. Figure 2 shows a schematic representation of this model.

ϕ_{user} allows the user and item embeddings to interact in a similar way to (He et al., 2017). ϕ_{item} is simply the representation vector produced by

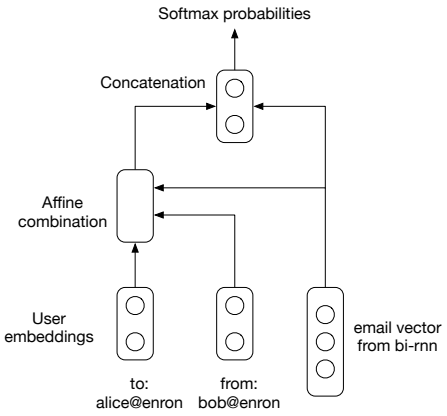

Figure 2: Schematic diagram of the sender-recipient interaction model.

the sequence model. The interaction prediction combines both of these features, and can use information from both. For full generality, the item feature ϕ_{item} could also be a function of p_a and p_b, as it is in (He et al., 2017). We leave this as a topic for future work.

This model assumes that there is only one recipient a. If an email has multiple "to" addresses, we arbitrarily choose the first one to be the recipient a. See section 7 for discussion on how this can be extended to accommodate multiple recipients.

4 Dataset

The dataset is derived from the Enron email corpus (https://www.cs.cmu.edu/~enron/), which contains over 1.5 million emails sent by 152 users at Enron. The dataset was originally made public by the Federal Energy Regulatory Commission during its investigation of the Enron bankruptcy scandal.

The Enron email dataset is not explicitly categorized into "no action taken" and "action taken" classes. However, "reply" and "forward" email chains appear as a single email example, and this can be used to categorize emails as "no action taken" and "action taken." We wrote a preprocessing script that uses regular expressions to detect whether an email contains a reply or forward chain. If so, the email extracts only the portion of the email after the "reply" or "forward" tag and labels the email as "action taken". Otherwise, the script takes the entire email text and labels it as "no action taken". The script also extracts the sender, the recipient(s), and other pieces of metadata such as the subject line and message ID. If the string parsing or tokenization of an example fails, the example is skipped.

The preprocessing script parsed roughly 40% of the 1.5 million emails into training examples. These examples were split into training, development, and testing datasets in an 80%-10%-10% ratio. Table 1 shows the number of examples and the class distribution of the training dataset.

Figure 3 shows the distribution of the lengths of a subset of 32000 emails from the training dataset. In this case, the length of an email is the number of tokens in its body. Notice that the distribution is long-tailed. Half of the emails have a length of less than 119, and three quarters have a length less than 271. However, 512 email, or 1.6% of the subset, had a length of greater than 2000. This long-tailed behavior makes it more difficult to train sequence models, as the models need to be able to handle both very short and very long emails.

Table 1: Class distribution of the training dataset

No Response	Action taken	Total
327,038	158,961	485,999
67.29%	32.71%	

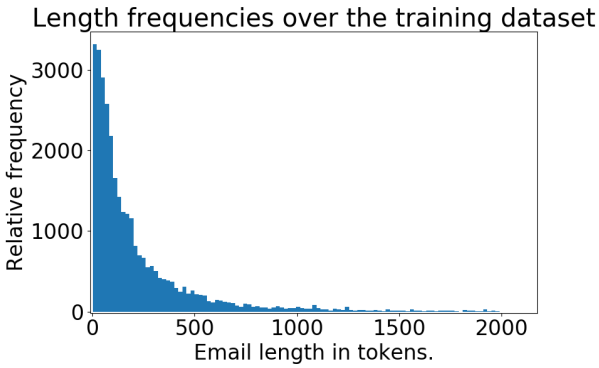

Figure 3: Distribution of email lengths over a subset of the training dataset

5 Experiments

The source code for the experiments presented here can be found at:

https://github.com/tleavitt/emailresponsemodel

5.1 Training configuration

We trained all models using gradient backpropagation on a cross-entropy loss function. We used the Adam optimization algorithm (Kingma and Ba, 2014) with learning rates between `1e-4` and `1e-3`. We used the 20,000 most common words in the training set as the model vocabulary, and used 50-dimensional GloVe vectors (Pennington et al., 2014) as the word embeddings. We trained using batches of 32 examples. We trained the linear model for 10,000 batches (roughly half of a data epoch), and LSTM models for 50,000 batches (roughly 3.3 data epochs). We were unable to obtain a GPU for training, and used a 4-core CPU machine instead. Training the linear model took around 5 minutes, and training the LSTM models took roughly 5.5 hours.

We found that an LSTM hidden dimension of 50 and an attention context size of 35 yielded good results and reasonable training times. The LSTMs were regularized by applying dropout to the input and state vectors. That is, at timestep t, the input x_t is replaced with $x'_t = m_1 \circ x_t$, and the previous state h_{t-1} is replaced with $h'_{t-1} = m_2 \circ h_{t-1}$, where m_1 and m_2 are Bernoulli boolean masks. We set $P(m_1[i] = 1) = 0.8$ and $P(m_2[i] = 1) = 0.5$ during training, and set both to 1 during validation.

We chose the model dimensions, learning rate, and dropout probabilities using a random grid search.

5.2 Results

Table 2 shows the precision, recall, and f1-score for classifiers trained in this experiment. [1]

The bi-directional LSTM with sender-recipient interaction (Bi-RNN w/ SRI) achieved the highest F1 score (77.02) and precision (81.37). The bi-directional LSTM (Bi-RNN) achieved the highest recall (77.86).

These scores are an improvement over previously reported scores for email reply prediction tasks of similar scope: namely, the 0.72 AUROC score reported in (Yang et al., 2017) and the 67.0 F1 score reported in (Dredze et al., 2008).

[1] for a description of macro-averaging, see the scikit-learn documentation.

Table 2: Performance statistics, macro-averaged over both classes. Statistics are reported as percentages.

Model	Precision	Recall	F1
Random Classifier	49.89	49.87	48.25
Linear LogReg	69.39	61.36	61.72
Bi-RNN	73.87	**77.86**	74.66
Bi-RNN w/ SRI	**81.37**	75.01	**77.02**
(Yang et al., 2017)	-	-	72.0
(Dredze et al., 2008)	73.0	-	67.0

6 Discussion

The models presented here improve upon prior F1 scores for the task of email response prediction. The bi-directional RNN model we use achieves state-of-the art results, and the sender-recipient interaction term further improves the model's score.

The difference between the plain Bi-RNN model and the Bi-RNN model with sender-recipient interaction becomes more apparent when looking at performance on just the minority class. Table 3 shows the precision, recall, and F1 scores for these models on the "Action taken" class. Here, the Bi-RNN w/ SRI model outperforms the Bi-RNN model by more than 5 percentage points on all three metrics.

Table 3: Performance statistics of the top two models for the minority class, "Action taken".

Model	Precision	Recall	F1
Bi-RNN	64.95	60.06	62.41
Bi-RNN w/ SRI	**81.95**	**65.27**	**68.19**

In the application of email recommendation, the performance on this class is more important; in particular the recall for the "Action taken" class is particularly important if the goal is to show users the important or anomalous emails currently

in their inbox.

It is unclear if a recall of 65.27 on the "Action taken" class is enough to power a real email recommendation application. If the application only recommends 65 of every 100 emails a user responds to, the user will still have to dig through their inbox to find the 35 emails that the model missed. Better performance may be needed for an application like this to be useful.

6.1 Attention Visualization and Interpretability

It is possible to visualize the attention weights of the LSTM on individual sentences, which can give insights into the words that the model is focusing on to make predictions. One such visualization is shown in figure 4. The model correctly classifies the email "I heard Pat Radford is leaving, and you know Bob Carter is leaving, right?" as "No response." The model seems to put a lot of attention on the word "leaving;" the two instances of "leaving" are the highest-scoring words in the email.

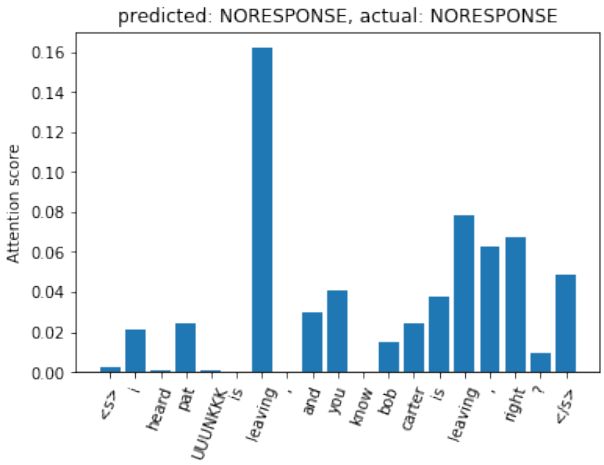

Figure 4: Attention weights for the bi-directional RNN model on a sample from the development set.

This example is noteworthy because the text intuitively seems to merit a response, but the model correctly classifies it as "no response." Curiously, the model seems to put very little attention on the question mark at the end of the sentence, which intuitively should be a strong indicator that the email needs a response.

This example provides qualitative evidence that even though the model performs reasonably well (in fact, better than a human on this particular example), the model's behavior decision process is not easily interpretable. This is a general characteristic of deep neural networks.

7 Future work

The grid search we used to select model hyperparameters was far from exhaustive. Model performance may be improved by a more thorough evaluation of hyperparameters. Furthermore, modifications to the model design may also improve performance. For example, one could try using a GRU instead of an LSTM, or try using a different attention scheme.

The sender-recipient interaction model described in section 3.2 models the interaction probability using sender-recipient features ϕ_{user} and email text features ϕ_{item}. Per equation (2) in section 3.2, the item features are simply the document vector produced by the email sequence model. For full generality, the item feature ϕ_{item} could also be a function of the sender and recipient vectors, as it is in (He et al., 2017). This will increase model complexity but could improve model performance by allowing for more complicated interactions between user information and the email text.

The sender-recipient model also assumes that there is exactly one recipient of every email, which is not true in practice: emails may have multiple "to" addresses or "cc" recipients. In fact, majority of emails in the Enron corpus have multiple recipients.

To extend the model to accommodate multiple recipients, the recipient vector p_a will have to be modified to incorporate information on an arbitrary number of users. Perhaps the simplest way of doing this is to simply sum or average the embedding vectors of all recipients. If necessary, a sequence model like an RNN or CNN could also be used to combine multiple recipient embeddings into a single representation.

Having a human "gold standard" for the email

response prediction task would help put the problem's difficulty into perspective and contextualize the results presented here. The author is hopeful that machine learning models can surpass human performance on this task; having a concrete performance benchmark could aid in model design.

References

Philip Bramsen, Martha Escobar-Molano, Ami Patel, and Rafael Alonso. 2011. Extracting social power relationships from natural language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 773–782, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mark Dredze, Tova Brooks, Josh Carroll, Joshua Magarick, John Blitzer, and Fernando Pereira. 2008. Intelligent email: Reply and attachment prediction.

L. Eugene and I. Caswell. 2015. Making a manageable email experience with deep learning.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 173–182, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Shikun Zhou Taiwo Ayodele and Rinat Khusainov. 2009. Email reply prediction: A machine learning approach. `http://www.iaeng.org/publication/WCE2009/WCE2009_pp31-36.pdf`.

Liu Yang, Susan T. Dumais, Paul N. Bennett, and Ahmed Hassan Awadallah. 2017. Characterizing and predicting enterprise email reply behavior. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 235–244, New York, NY, USA. ACM.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.