# Predicting the Likelihood of Response in a Messaging Application

Kevin Shin, Tushar Paul
{kevshin, aritpaul}@stanford.edu

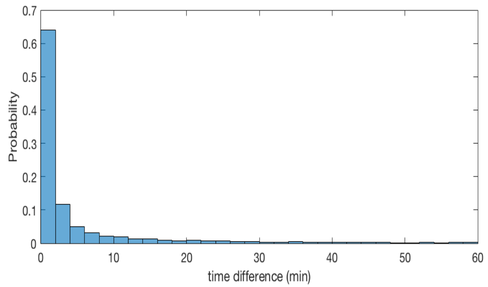
## Motivation

Group messaging services send users notifications when they get messages. We aim to classify these messages by the user's response likelihood. The service could then only send notifications that the user is likely to respond to.

## Data Source

We used data from one Facebook Messenger group that both group members are in. This group has 27 users with over 10,000 messages.

We had different labelings per user. All messages sent 20 min before messages from that user was labelled "responded to". This threshold came from $2\sigma$ of the message time difference distribution:




## Features

For each user, we extracted features they may consider when deciding whether to respond:

```
[Number, Length,
Type, Sender, Day, Hour]
```

The **Message Number** is in the context of a particular conversation, and the **Message Type** is one of [Question, Link, Comment]. We also considered the text itself. For LSTM we passed in a vector representation of the first 20 words and for all other models we passed a bag of words.

## Learning Models

**Naïve Bayes:**

We predict based on the result of:

$$\arg\max_y p(x|y)p(y)$$

Bernoulli Naïve Bayes:

Assume features are Multivariate Bernoulli [1]

$$p(x_i|y) = p(i|y)x_i + (1 - p(i|y))(1 - x_i)$$

Gaussian Naïve Bayes:

Assume features are Gaussian [1]

$$p(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

**Support Vector Machine (SVM):**

SVM solves the following to maximize the margin and minimize the training error

$$\min_{\gamma,w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i \text{ s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0,\ i = 1, \cdots, m$$

We used SVM with the RBF Kernel below:

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$

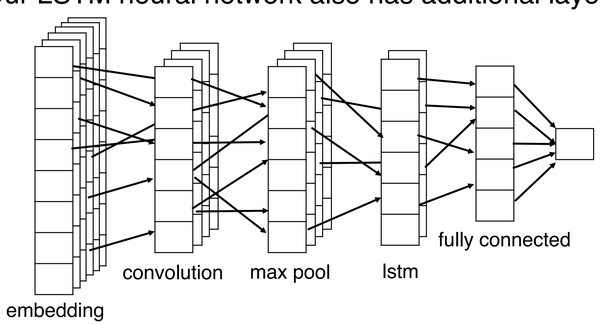
**Logistic Regression:**

The hypothesis function is:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

**Long Short Term Memory (LSTM):**

Our LSTM neural network also has additional layers.




Each LSTM cell computes the following [2]:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$h_t = o_t * \tanh(C_t)$$

## Experimental Results

We focused on one of the most active members in the group. During our testing, we saw that users had similar error, FP and FN rates, but this user had the best F1 scores:

| | Train | | Test | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model | Error | F1 Score | Error | F1 Score | Precision | Recall | FP Rate | FN Rate |
| Bernoulli NB | 0.394 | 0.022 | 0.403 | 0.014 | **0.769** | 0.007 | **0.001** | 0.499 |
| Gaussian NB | 0.594 | 0.573 | 0.593 | 0.577 | 0.406 | **0.999** | 0.499 | **0.001** |
| Logistic Regression | 0.294 | 0.617 | 0.296 | 0.622 | 0.643 | 0.602 | 0.185 | 0.285 |
| SVM w/ RBF Kernel | 0.325 | 0.654 | 0.326 | 0.654 | 0.573 | 0.763 | 0.279 | 0.192 |
| LSTM Neural Net | **0.244** | **0.729** | **0.292** | **0.682** | 0.609 | 0.774 | 0.252 | 0.185 |

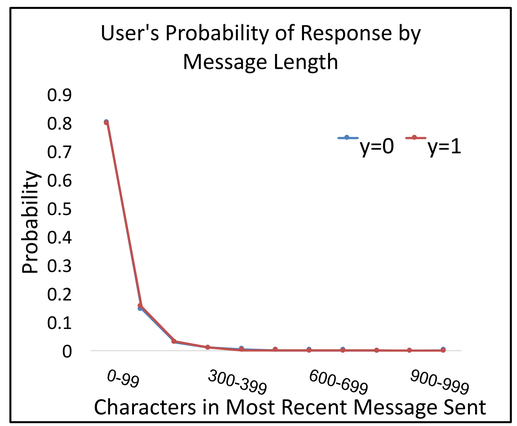
We used a 70/30 train/test split of our data, which came out to 7900/3400 examples. We used **cross validation** and **feature selection** to pick the best parameters and features for each model.

## Discussion

While Naïve Bayes failed with our imbalanced data, we mitigated the effects in our other models by weighting [1] examples inversely proportional to their class frequency:

$$w_{C_1} = \frac{\text{num train examples}}{\text{num train examples in class } C_1}$$

Looking at all our evaluation metrics, **LSTM is our best model** for this problem. The main advantage LSTM has is that it makes better use of contents of the message than our other models. Interestingly, the **message length either hurt or didn't change the performance** of our models. The graph on the right shows that our data corroborates this result. We also found that users with less imbalanced classes led to higher F1 scores. Finally, our LSTM model had close to state of the art accuracy [2].




## Future Plans

In Lee and Demoncourt's research with classifying short-text, convolutional neural networks proved to give great results as well which gives us another model to implement [1]. We could also go deeper into Natural Language Processing and find other message types, named entities, topics, etc. We could also try testing additional group chats and users. Finally, it would also be interesting to try and implement the classifiers we built into a real notification filter by building a basic chat application.

## References

[1] scikit-learn developers 2016, "scikit-learn: Machine learning in python," 2010. Web. `http://scikit-learn.org/stable/documentation.html`.
[2] Lee, Ji Young, and Franck Dernoncourt. "Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks." *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2016): n. pag. Web.