Introduction

My friends and I have a group chat on Facebook that we started two years ago where we talk about academic and social activities. Over the lifespan of the chat, it have accumulated over 18,000 messages. My model generates a typical message for each one of the people in the group chat, and given a message, it predicts who sent it.

One of our traditions is to hold a barbecue each 4th of July, but since we all graduate next semester, it is not possible. Instead, we are going to move the barbecue to sometime before May. At this party, we will review the group chat as a reflection on our time as friends. This model will provide a means to do so. Specifically, the model will generate a typical message for each person, allowing us to see what each person generally talks about, and given a message, specifically outlandish ones, it will guess who was most likely to have sent it.

Approach

Facebook offers the ability to download a copy of all of the information associated with an account. This download includes html code for each message thread. With the use of Text Wrangler's string replacement tool, I was able to get each message on one line along with the sender of the message and the date it was sent. Something I did not expect to encounter was over a thousand messages with no contents. I believe the messages were ones whose contents were only gifs; these had to be deleted from the data.

In generating a typical message for everyone, I built a Markov model with a history of one word for each person. I then generated a word sequence corresponding to the average length of a person's message that greedily maximized the transition probability for each word pair. For example, if there was a person name John in the chat, and John's average message length was two, his typical message would be the word John most likely starts messages with and then the word that is most likely to follow this most likely first word.

In predicting who sent a message, I used a recurrent neural network that takes word embeddings as an input. The word embeddings I used were the same ones Michael Capizzi used in his recurrent neural network presentation (https://levyomer.files.wordpress.com/2014/04/dependency-based-word-embeddings-acl-2014.pdf). For each message, each word is classified as belonging to a certain user. Then to classify who sent the message, the user with the most word classifications in the message was selected.

Code

In order to train the Markov model, transitions probabilities for each user were calculated. These probability calculations were done by simply partitioning the entire dataset by user. For each user, the transition probability of two words in their vocabulary, t_i and t_{i-1}, was calculated as:

$$P(t_{i-1} \mid t_i) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

Afterwards, the average message length of each user was calculated using a simple mean calculation. Finally, to generate a typical message, for each user, the code generated the sequence:
$\{W_1 = argmax(P(w_i|START)), W_2 = argmax(P(w_i|W_1)), \ldots, W_n = argmax(P(w_i|W_{n-1}))\}$
where n is the average message length for a given user and w_i is a word in the user's vocabulary.

In order to load the pretrained word embeddings, I wrote a python script that parsed each word and its 300-dimensional word embedding and then created a dictionary from words to embeddings. In order to build an embedding look up for Dynet, each word in the combined vocabulary of all users was mapped to an index in a list where the entry at that index is the word embedding of the corresponding word. The word embedding (blue) then fed into a hidden layer (green) of size 200, which in turn fed into another layer of size 6, the number or people in the chat (yellow). Finally, I added my own layer (red) of size one. This mapping of size one simply counted the number of word classifications each user had in a given message, and then picked the user with the most word classifications in the message.

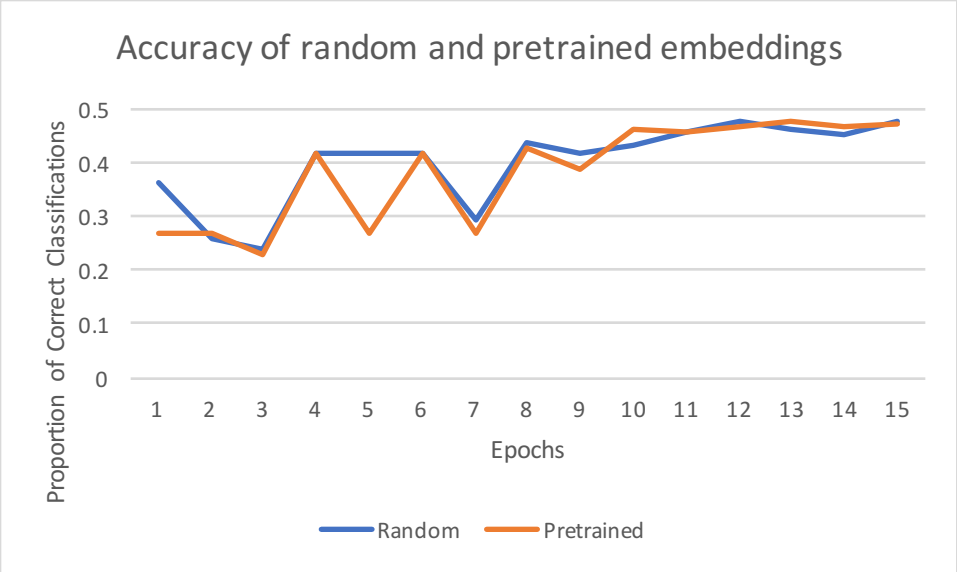
Results

The typical message for each person is the following:

```
Jake Croft : I was having writers block, was having writers block
Daniel Osorio : I have to go to go to
Emanuel Bustamante : I don't know what the hookah lounge
Jose Celaya : I don't know, I don't
Olivia Gorushi : I don't know what I don't know what I don't
Paul Acosta Estrada : I have a lot of the same
```

Note that everyone's messages except mine and Paul's seem to get stuck in a loop of words that make up a phrase. A possible explanation for these cycles is the limited window size of one used to generate transition probabilities.

Disappointed to have only seen less than 50% accuracy using random word embeddings, I decided to try pretrained embeddings. However, the pretrained embeddings were just as inaccurate:



For experience's sake, I used pretrained word embeddings, and achieved a correct classification rate of 47%. Considering that Olivia sent over half of the messages, I could have implemented a better classifier by just choosing Olivia every time. Due to this lack of accuracy, my model will not be receiving an invitation to the barbecue this year.

Perhaps a maximum entropy Markov model would have classified better with some possible features being mentions of other people, length, response time, and time the message was sent. Additionally, some sort of tagging could have been implemented to incorporate emojis and GIFs.