Emoji Recommendation

By William Dahl

Introduction:

The goal of this project is to recommend an emoji for someone to send along with their text message. Adding this feature to any messaging app will result in better usability and a better user experience causing more people to want to use it. A challenge for this project is analysis the vast differences in varying meaning of text as used my humans to communicate. Working around things such as satire, text message shorthand, and the use of punctuation to accent a persons emotions can be difficult. Cultural changes in language can also provide a hurdle to analyzing text dialog between humans.

Related Work:

Within recent time, messages apps have begun to recommend emoji use. These platforms include apps such as imessage, Facebook Mssenger, Whats-app, and Snapchat. The paper “Neural Emoji Recommendation in Dialog Systems” by Ruobing Xie, Zhiyuan Liu, Rui Yan, and Maosong Sun of the National Lab for Information Science and Technology, Tsinghua University, Beijing, China and the Natural Language Processing Department, Baidu Inc., China Try to use Neural networks as a method for classifying dialog between two people and then recommend an emoji. The app Danjo uses Deep learning techniques to recommend the top 4 emojis to use given a message typed by the user, the app even extends this to gifs and memes as well. DeepMoji is an open source AI project where when given a word message will recommend 5 emojis to use in its place. It even has the ability to understand sarcasm and has access to millions of emojis.

Method:

At its roots, this is a semantics analysis problem.

My first method is to analyses how often different sayings appear in a given message and then use that to predict the semantics of the text. I received my data from the DeepMoji git repository. The Data is orginised with a vector of length 7. Each index within the vector represents a different emotion such as [joy, fear, anger, sadness, disgust, shame, guilt]. And then followed by the respective message.

E.x:

[1. 0. 0. 0. 0. 0. 0.] During the period of falling in love, each time that we met and especially when we had not met for a long time.

The above example has a joy sentiment.

My first step was to separate the messages from their labels so that I could treat the labels as a matrix as well as divide my data set into my training data and my testing data. I did this by doing a 80% and 20% split respectively.

The next step was to create n-grams. These are a combination of up to n words in a message.

E.x:

In the message “I love you” a 1-gram would be [‘i’, ‘love’, ‘you]. A 2-gram would be [‘i love’, ‘love you’] and 3-gram would be [‘i love you’].

For each message I created n grams from 1 to 4 in order to account for negation such as ‘happy’ versus ‘not happy’ and ‘love you’ versus ‘don't love you’.

Next was to classify the n-grams. I did this by analyzing the mode of the occurrences of a phrase or word trough out the training data and it’s label.

E.x:

the class distribution for the phrase “happy today” may look like this
[0,1,2,3,4,5,6,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,3,0,3,3,3,0,0,0,0,0,0,0,0,0,0,0,0,0,]

here the mode of the class distribution is 0 which maps to the feeling of joy.

Next is to classify a given message. I accomplish this by breaking the message into n-grams of size 1 through 4. I then compare the n-grams from the message to my already classified ones and create a distribution of possible classes to the message. I also apply a larger weight to the longer n-grams as clearly ‘not happy’ should weight more than ‘happy’. I accomplish this by putting the class for the phrase into the class distribution n times.

E.x:

“i am so happy” will have a class of 0 (joy) and is a 4-gram thus 0 will be written into the class distribution 4 times

I then take the mode of the class distribution and use that as my predicted value for the class of the message.

Results:

My initial results using this method is a 100% accuracy when applied to the training data and a 49.9% (373 out of 784) accuracy when applied to my testing data. This implies that my model is probably over fit. Also, My attempt to consider negation on adjectives appears not to have been successfully as phrases like “not happy” and “not sad” still map to joy and sadness respectively. A possible fix to this could be to display not just the number one recommended emoji but the maybe the top 3 as most already implemented solutions do.

Next Steps:

The next steps that I would like to take with this project is to implement the use of linear SVC to classify the data and compare it’s results to the results of the mode analysis. I also need to create the mapping for the semantics and their respective emoji. I would also like to create a user interface that dynamically shows the recommended emojis as the user types. Also, if time permeating I wold like to apply neural networks and deep learning principals in an attempt to get the correct recommendation rate higher. The run time for the program is also very high, I plan on lowering it by loading my classified n-grams into a file and having the model use that for classification so that the n-grams are not classified every time the program is run. There are also some bugs with the creating of my data files so those too need to be fixed so that I don't need to go in a manually remove blank lines.

Data Source from:

https://github.com/bfelbo/DeepMoji/tree/master/data