**TraitSpot: Improving Personality Detection using Social Relationships in Latent Neural Embedding Space**

**Abstract**

The comments/reviews posted by friends in social media could be able to reflect the personality of a person. In this study we analyze how reviews posted by a user’s friends in social media can be used to determine the personality of a user. Specifically, we want to know whether we can improve personality detection using social relationship and to what extent we can predict the personality traits of each users with their friend’s reviews. The study is mainly focused on determining the nature of users who have posted less reviews in social media. In our main analysis we created a user-friend-review benchmark dataset for personality prediction using (review data from 1054 users and their friends) Yelp dataset for 1054 users, whose personality then we estimated using three different datasets. Our results from a preliminary NN model shows significant difference compared to the baseline model

**Introduction**

We are able to determine a user’s personality through social media by the contents posted by them as text, images, likes, dislikes or through interaction with others. The way in which users express themselves is a type of behavior usually determined by differences in their psychologic traits. Using large data sets of users and their online behaviors, recent studies have managed to successfully build models to predict a wide range of user traits such as age, gender, occupation, political orientation and personality. Out of these traits’ personality is one of the interesting characteristics that can be considered for adaptation purposes. With the information about one’s personality we can determine or at least have a hint about how she would react when encountering different situations.

Detecting the personality of a person using social relationship is a rarely studied online behavior. This mainly comes into picture in the scenario where there are users who do not express themselves much` in social media and have very less posts posted by them. Detecting personality of such users with just their data wouldn’t be much accurate. In this case we are trying to determine whether the posts made by her friends can be used to detect the personality of that person. This study is made in the assumption that people with similar interests most likely tend to be friends with each other in social media and there is a possibility their personality can match.

Personality is typically formally described in terms of the following Big Five personality traits:

* Openness - Includes traits like having wide interests and being imaginative and insightful.
* Conscientiousness - Includes traits like organized, thorough, and planful.
* Extraversion - Includes traits like talkative, energetic, and assertive.
* Agreeableness - Includes traits like sympathetic, kind, and affectionate.
* Neuroticism - Includes traits like tense, moody, and anxious.

Our main analysis uses a dataset of over 1054 Yelp users, their reviews and their friend’s reviews. Personality traits for each user was estimated as ground truth values using personality insights from IBM. Only the user’s own reviews were used as text input for ground truth deduction. We create 3 different datasets and train a neural network model with each dataset. The model outcome is personality trait score for each user with the different datasets. We also calculate the root mean squared error comparing with the ground truth for each dataset. With our analysis, we aim to present a model to improve personality detection of users with limited social footprint using the footprint of social friends

**Data preprocessing**

Yelp datasets (<https://www.yelp.com/dataset>) were used to conduct our experiment. We used only the review and user JSON files because our requirement was to acquire data regarding users their friends and posts/reviews by both users and friends. Out of 1,637,138 users filtered 1054 target users with each user having more than 100 friends and both user and friends having more than 50 reviews. Created mini python programs to execute the below steps for our initial filtering process.

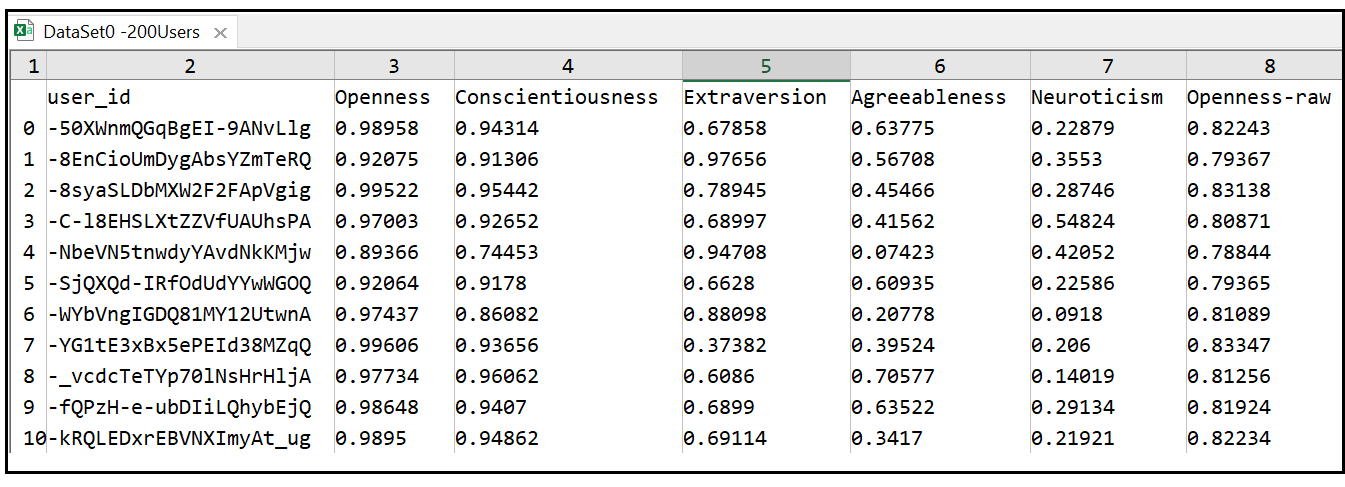
* In review file extracted the columns user\_id and text (review text) and removed all the users who has less than 50 reviews.
* In user file extracted columns user\_id and friends and removed all the user\_ids which are not in the extracted user\_id list from review file.
* Removed all the user\_ids who has less than 100 friends from previous step extracted data.
* Removed the friends for each user who has less than 50 reviews or no reviews.
* Again, removed all the user\_ids who has less than 100 friends.
* The above steps resulted a list of 1054 target users with each user having more than 100 friends and both target users and friends having more than 50 reviews.
* Consolidated all the reviews for each target users.
* Created a separate list for friends and consolidated 10 reviews for each friend (Initial plan was to use all the reviews but due to memory constraint reduced to 10 per friend).
* Combined the friends reviews for each target user (10 review per friend according to previous step).
* Target users had an average of 260 reviews and 220 friends with each friend having around 10 reviews.

**Datasets**

We created the below four final datasets after the initial data pre-processing and filtering.

**Dataset 0 (Ground truth):**

The requirement of this dataset is to calculate the big five user personality trait scores for each selected target user. Data used is 1054 target users and consolidated reviews of each target user where each user has an average of 260 reviews. IBM Watson Personality Insights API was used for ground truth deduction and consolidated review for each user was used as text input to the API. We made around 1054 API calls to obtain the ground truth values for 1054 target users. The Initial output for each API call was a JSON file containing all the personality data of the user. Extracted the raw and normal value of each of the five-personality trait from the JSON file and saved as dataset 0. JSON file for each user was saved separately for future purposes.



**Dataset 1 (Users only + limited social footprints):**

This dataset contains each target user and her selected 3 reviews. Selected 3 reviews for each target user and combined them as review data for each target users.

**Dataset 2 (Users only + sufficient social footprints):**

This dataset contains each target user and ALL her own reviews.

**Dataset 3 (Users + limited social footprints + friends’ footprints):**

This dataset contains each target user and her selected 3 reviews, together with ALL the reviews posted by all her friends. Selected 10 reviews for each friend and combined them. (Used only 10 review per friend due to memory constraint). Consolidated reviews from all the friends for each target user. Combined 3 selected reviews from target user and consolidated friends reviews of each target user and saved as Dataset 3.

Eg: A target user with 100 friends would have 3 + (100\*10) =1003 reviews combined

**Implementation**

**Feature Extraction**

Text data requires special preparation before we can start using it for predictive modeling. The text must be parsed to remove words, called tokenization. Then the words need to be encoded as integers or floating-point values for use as input to a machine learning algorithm, called feature extraction (or vectorization). The scikit-learn library has some tools to perform both tokenization and feature extraction of the text data. Using one of those called the TfIdfVectorizer the text data for each user was converted to word frequency vectors (TFIDF vectors). TfIdfVectorizer transforms text into a sparse matrix where rows are text and columns are words, and values are the tf-idf values. Tf-idf in general is a method used to weigh a keyword in any content and assign the importance to that keyword based on the number of times it appears in the document. The parameters for TfIdfVectorizer were set to the default values.

Some of the important default settings are below:

* max\_df = 1.0 : When building the vocabulary, it ignores the terms that have a document frequency strictly higher than the given threshold (corpus-specific stop words). If float, the parameter represents a proportion of documents.
* min\_df =1 : When building the vocabulary, it ignores the terms that have a document frequency strictly lower than the given threshold. This value is also called cut-off in the literature. Integer represents absolute counts.
* max\_features= none : Considers the maximum number of elements that is the whole corpus if the value is set to none

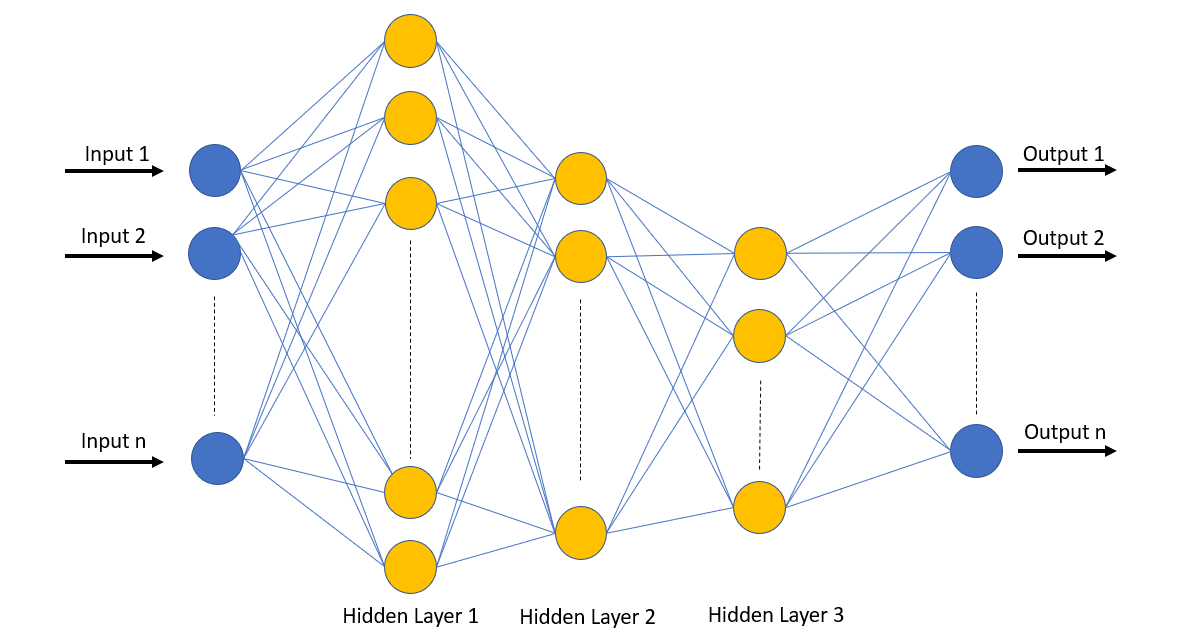
After tokenization the fit\_transform method from sklearn is called to learn a vocabulary from one or more documents and encode each element as a vector, this process is called the vectorization. The text data for users are transformed into ‘200 x Max no. of elements’ sparse array in the form of ‘numpy.ndarray’. Where each row denotes each user and each column denotes each element from the text data. The number of rows is 200 because we considered only the first 200 users out of 1054 target users for our experiment. The scores are normalized to values between 0 and 1 and the encoded document vectors can then be used directly with most machine learning algorithms.

**Model Implementation**

A Neural network model was created using sklearn. The data is split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later. We have the test dataset in order to test our model’s prediction on this subset. Imported the train\_test\_split method from Sklearn, sub-library model\_selection to split the data to training and test sets. The input parameters for the train\_test\_split method are the vector output from feature extraction, targets users ground truth values for a single personality trait from data set 0 and the test size set to 0.2.

The test\_size=0.2 indicates the percentage of the data that should be held over for testing. It’s usually around 80/20 or 70/30. I focused on working with only a single personality trait that is ‘Agreeableness’ because it had the most distributed ground truth values between users as per the histogram. The output parameters are x\_train, x\_test, y\_train, y\_test where x\_train is the 80% data from feature extraction and x\_test is the remaining 20% data, y\_train is 80% data from ground truth values and y\_test is remaining 20%.

Since the data is in the form of numpy.ndarray with real numbers stored in variables x\_train and y\_train, we trained the model using the method MLPRegressor from sklearn.neural network. MLPRegressor implements a multilayer perceptron that  [trains](http://www.gabormelli.com/RKB/index.php?title=train&action=edit&redlink=1) using  [backpropagation](http://www.gabormelli.com/RKB/backpropagation) with no [activation function](http://www.gabormelli.com/RKB/activation_function) in the [output layer](http://www.gabormelli.com/RKB/output_layer), which can also be seen as using the [identity function](http://www.gabormelli.com/RKB/identity_function) as [activation function](http://www.gabormelli.com/RKB/activation_function). Therefore, it uses the [square error](http://www.gabormelli.com/RKB/square_error) as the [loss function](http://www.gabormelli.com/RKB/loss_function), and the [output](http://www.gabormelli.com/RKB/output) is a set of [continuous values](http://www.gabormelli.com/RKB/continuous_value). **Multi-layer Perceptron (MLP)** is a supervised learning algorithm that learns a function *f(⋅):Rm→Ro* by training on a dataset, where *m* is the number of dimensions for input and*o* is the number of dimensions for output. Given a set of features *X*=*x1,x2,...,xm* and a target *y*, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers.  We used 3 hidden layers with sizes 1000,200 and 100 and the max\_iter was set to 5000 so that the training converges with a greater number of iterations.



**3-Hidden layer Multi-layer perceptron**

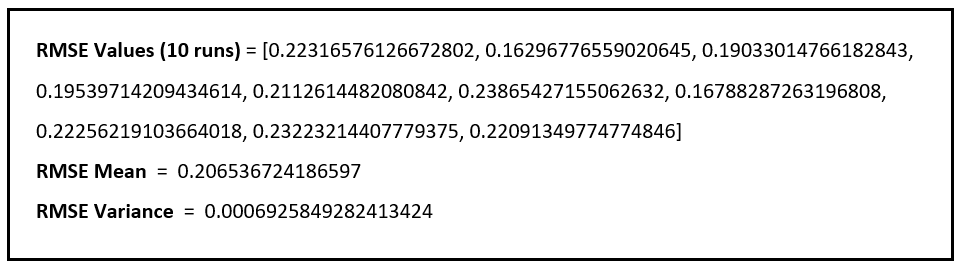
Next executed the fit() function which determines the parameters of the neural network. Only after calling fit, the neural network is successfully initialized. We fit the model using x\_train and y\_train as input parameters and it returned a trained MLP model. After the fit function at last we predict using the multi-layer perceptron. The input parameter for the predict function is x\_test which is the 20% data from feature extraction and expected output is the predicted values. Then we compared the predicted values against y\_test( 20% data from the ground truth values) and calculated the root mean squared error of the model. Each time I ran the model there was a slight difference in the RMSE score. So, I had to run it multiple times and calculate the mean and variance as model results.

We created a model with the above explained functions to implement feature extraction and neural network model. The model was executed separately for datasets 1, 2 and 3 expecting different results for each dataset. We tested the model only with 200 users out of 1054 users due to memory constraints. The model was tested only for a single personality trait that is “agreeableness”. Out of the five personality traits we chose agreeableness because it had the most distributed ground truth values as per dataset 0.

**Results and evaluation**

**Dataset 1**

We got the below results for dataset1 after running model implementation in loop for 10 times to obtain the mean and variance or RMSE score. We had to get the mean value of RMSE score due to the slight difference in values after each time we ran as you can see with below results.

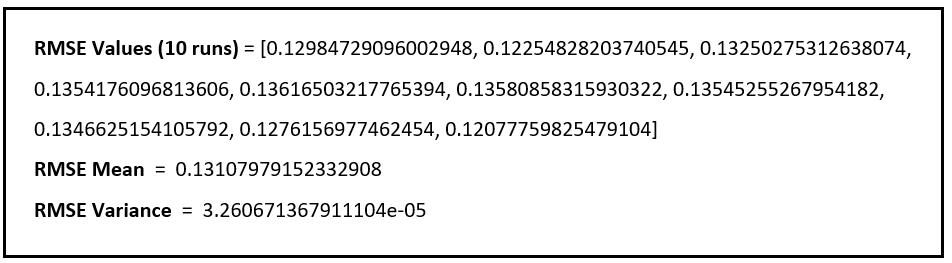


**Dataset 1 Results**

RMSEMean value is the mean squared error between predicted agreeableness trait scores and agreeableness ground truth values for 20% of users (10 users). We can say that RMSE values are quite static among multiple runs because the variance is very less. Dataset1 has a mean squared error around 0.207 where their accuracy compared to the ground truth value is good but not the best. The Dataset1 has only 3 reviews per users so our expectation is the trait accuracy to be not the best out of all 3 datasets.

**Dataset 2**

We got the below results for dataset2 after running model implementation as same as dataset1.

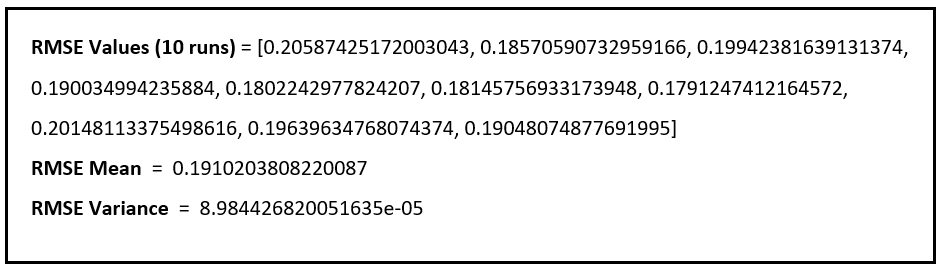


**Dataset 2 results**

The RMSE values are more constant compared to dataset1 because the variance is very less. Dataset2 has a mean squared error around 0.131 where their accuracy compared to the ground truth value is the best out of all datasets. That is because Dataset2 has all their own reviews consolidated where each user has an average of 260 reviews, so our expectation is the trait accuracy to be the best out of all 3 datasets.

**Dataset 3**

We got the below results for dataset3 after running model implementation as same as other datasets.



**Dataset 3 results**

Dataset3 has a mean squared error around 0.191 where their accuracy compared to the ground truth value is the better than dataset1 but lesser than dataset2. Dataset3 has user’s limited reviews plus consolidated reviews of all friends, so our expectation is the trait accuracy to be better than dataset1.

It’s expected that dataset2 model will give the most accurate results and looking at the implementation results we can see that its results outperforms all. That is because the prediction is direct only when their own posts are involved and also in large quantity (more content increases the accuracy of the models). We can say that the next better results are from dataset3 and we can also see that it outperforms the baseline model (dataset1 model) for the personality trait “agreeableness”. This is a proof that as per our analysis we can determine the personality trait of users with limited social prints using their friend’s social media posts.

**Limitations and Future work**

We had a few limitations during data processing and model implementation due to memory constraints. I initially planned on consolidating all the reviews of each friends which ended up in a very large data so reduced the review count as 10 per friend. The next cut was reducing the number of target users to 200 from 1054 users because was not able to run my model for datasets 2 and 3 on a computer with 16GB RAM. The results would have been a little better if we had used the entire data for our implementation.

In Future can work on improving the model to give better results with dataset3 compared to dataset1 by using the entire data for model implementation. We can do a detailed analysis on how close the friends’ personality scores are to each user, compared with her non-friends using the dataset 0. We can also work extracting the LIWC features for datasets and train a version2 neural network model using LIWC features.

**Conclusion**

Our main analysis was to determine whether reviews posted by a user’s friends in social media can be used to determine the personality of a user. We wanted to know whether we can improve personality detection using social relationship and to what extent we can predict the personality traits of each users with their friend’s reviews. We gathered the review data of users and friends from Yelp data set. Filtered around 1054 target users from the main dataset with each user having more than 100 friends and both user and friends having more than 50 reviews. Using this data created 4 different datasets. Dataset 0 was created using IBM personality insights API which contains the ground truth values. Datasets 1, 2 and 3 were implemented using TFIDF vectorization and neural network model. The RMSE scores of each model was compared with each other. The results showed that the dataset 3 model outperforms the dataset 1 model. We can conclude that for people with limited social footprints the posts made by their friends can be used for personality detection and it has a better accuracy rate compared to their own limited reviews.