TCSS 455 Intro to Machine Learning Social Media User Profiling

Khatra, Navdeep
Li, Tianyi
Zhou, Zebin
Team 12, Spring 2019
University of Washington, Tacoma

Student ID: 1726648
Student ID: 1827924
Student ID: 1667016

ABSTRACT

As more and more users are creating their own content on the web, there is a growing interest to mine this data for use in personalized information access services, recommender systems, tailored advertisements, and other applications that can benefit from personalization. Research in psychology has suggested that behavior and preferences of individuals can be explained to a great extent by age, gender and underlying psychological constructs (or so called personality traits). In addition to a myriad of applications in e-commerce, there is a growing interest of user profiling in digital text forensics as well [1].

The goal of this report is to build a system for automatic recognition of the age, gender, and personality of social media users. When given as input the text status updates, profile picture and ĂIJlikesĂİ of a social media user (relation), this system should return as output the age, gender and personality trait scores of that user [2].

KEYWORDS

Machine Learning, Social Media, Virtual Machine

1 INTRODUCTION

For this project, we used a large amount of training data from social media to build several machine learning models for profile the account users. A range of methods have been suggested to automatically predict the age, gender and personality trait scores (OCEAN) of the users from their contents, profiles and images produced by them. During the process, we utilized different sources of data and hence explored different methods of modelling. Approaches differ in terms of the data sets and machine learning algorithms used.

In the center of this project, we accept 3 kinds of sources as the inputs: texts (users' status updates), relations (users' "likes" on different post pages) and images (users' profile pictures). Our machine learning algorithms will process them and return an output as prediction to the ages, genders and personality trait scores of the testing social media users.

In order to achieve this goal, we created different models based on the various inputs datasets.

- With texts datasets, we build models genders, ages and OCEAN scores
- With relations datasets, we predicted genders and ages
- With user's profile image datasets, we created algorithms for genders and ages.

By the end, we picked out our best models (higher accuracy) and build a trial method for voting by majority.

2 DATASETS

A training dataset with 9500 labeled instances is provided. The dataset contains profile information of 9500 Facebook users, collected with the MyPersonality1application. The MyPersonality app was a popular Facebook app introduced in 2007. Users using the MyPersonality app took a standard Big Five Factor Model psychometric questionnaire and gave consent to record their responses and Facebook profile information.

The main dataset that will used in all of the model inputs are the users' profile information, called 'profile.csv'. It is a comma-separated values (csv) file with columns of 'userid', 'age', 'gender', and their personality scores for all 9500 users. In the 'gender' column, '0' denotes male and '1' denotes female.

All 9500 users declared in their Facebook profile that their language is English. Still, a small fraction of the status updates is not in English.

2.1 Texts

Data resources are a number of text files with the status updates of the users. All the status updates of a single user are combined into one file called '<userid>.txt'. There are 9500 such text files, one for each user.

2.2 Relations

The dataset is a csv file with rows of the columns 'userid', 'like_id'. Each row indicates which pages that a user have liked.

2.3 Images

A folder of profile pictures of the users were used, with the naming convention âĂIJuseridâĂİ.jpg. There are 9500 such images, one for each user.

2.4 LIWC

As mentioned above, we also have the opportunity to analyze users' personality trait scores. In this case, we give each user a score for their Openness, Conscientiousness, Extroversion, Agreeableness and Neuroticism, also called their OCEAN scores. Each of them are ranged within 0 to 5, up to 2 decimal places.

To predict users Ocean score, we have a csv file, called 'LIWC.csv', with 81 features in the table. Because this information is texts base, it is a piece of task that would work better with other texts inputs.

2.5 Training vs. Testing Datasets

To successfully predict the Gender, Age, Personality of the users using a machine learning model was challenging. We were given another folder of public testing data. In that folder, there was a csv file that had 334 people with their profile as columns.

Each of the team were also given a testing virtual machine for the instructors to run evaluations on the scripts, in which, there are also 1000 hidden users information as private testing data

3 METHODOLOGY

3.1 Texts

3.1.1 Gender Predictions. The first task our group could approach was to start the prediction on user genders with their texts as the source. The users were given a unique user ids in the profile.csv table. we thought to apply the Naive Bayes model on the data set. But in order to apply that model we needed a column named text which contains all the text that was uploaded by the user. All the text files were located in the different folder. In that folder there were 9500 txt files. Each text file was named the same as the user ids in the profile.csv folder. So, our first step was to think of an algorithm that help in merging all those text files into the profile.csv table.

The names of the text files indicated the user ids of the users. The names are in the format of 'xxxxx.txt', where 'xxxxx' are the the user ids and each user has their own file by adding a '.txt' by the end. Then the string left is the user id of the user that we have in the profile.csv table which is 'xxxxx.txt'. So, we created a column named text in profile.csv table and wrote some code to merge the text into that table. According to the algorithm, we did a for loop to make a text list where the user id and matching text is stored in accordance with each other. Then we merged the user text into the profile.csv file in accordance with matching user ids.

After that step, we had everything in one place that we needed to apply our machine learning model. Then the count vectorizer and Naive Bayes models were explored in detail. As we needed something that divides the text into vector, so that it could be further helpful to apply the probability formulae and figure out whether the males use this term more often than female or vice versa. The count vectorizer and Naive Bayes formed a good package together to calculate the gender of the user. So after configuring their functionality and stability our next step was to use these two methods together and create a model that predicts the gender of the user using count vectorizer and Naive Bayes.

To use the right two columns named Gender and Text from profile.csv file, we used a dataframe.loc functionality to get values of those two columns based of their labels. Once we had the right columns then we applied the count vectorizer on the text column and trained our model based of the gender of the text written by the users. Then we applied the Naive Bayes model on the model and fitted our gender and text training data on them. After successfully applying the model to 8000 training data instances the first 1500 were decided to be used as testing purposes.

After successfully deploying the model and upon successfully training the data, now we had to test the model. we had kept aside first 1500 examples from the dataset provided to use them for testing. Therefore, now we do prediction for those testing examples by using the model that we trained from 8000 examples. After doing the prediction we compared the predicted value to the actual value in the test examples and calculated the accuracy score from there. we was able to achieve 0.70 accuracy with that.

The next step was to apply this model to our virtual machine. we did the prediction of the data according to their text whether the user is a male or a female. All that prediction was stored in a temporary array. Then on the virtual machine each user's '.xml' file was produced where all the other information beside gender was hard-coded but the gender prediction was made using the Naive Bayes model. Now, the temp array was called and all the predicted genders were inserted into the '.xml' file. This came out to give me a 0.70 accuracy on virtual machine too. We successfully met the baseline with gender prediction of the users with 0.70 accuracy.

3.1.2 Age Predictions. The Second task was to do the age prediction of the users according to their texts as an source. We already had merged all the text into the profile.csv file which also had age and gender column in it. So, we had everything in place that was needed to deploy our model. As we needed something that divides the text into vector, so that it could be further helpful to apply the probability formulae and figure out whether the males use this term more often than female or vice versa. The count vectorizer and Naive Bayes formed a good package together to calculate the age of the user. So after configuring their functionality and stability our next step was to use these two methods together and create a model that predicts the gender of the user using count vectorizer and Naive Bayes.

Again, to use the right two columns named Age and Text from profile.csv file, we used a dataframe.loc functionality to get values of those two columns based of their labels. Once we had the right columns then we applied the count vectorizer on the text column and trained our model based of the age of the text written by the users. Then we applied the Naive Bayes model on the model and fitted our gender and text training data on them. After successfully applying the model to 8000 training data instances the first 1500 were decided to be used as testing purposes.

After successfully deploying the model and upon successfully training the data, now our next step was to test the model. we had kept aside first 1500 examples from the dataset provided to use them for testing. So, now we do prediction for those testing examples by using the model that we trained from 8000 examples. After doing the prediction we compared the predicted value to the actual value in the test examples and calculated the accuracy score from there. we was not able to achieve our accuracy more than the baseline. we just had a 0.59 accuracy which was same as baseline. Our next step was to figure out some other methods or models that might increase our accuracy.

The other model that came up to our mind was Logistic Regression. Applying this model on same dataset without changing the testing and training data this model helped us increase our accuracy to 0.63. But still was not satisfied with accuracy because it might again go down on virtual machine. Thus, our next step was to closely look up the data and apply some feature selection and study the data more in depth.

Upon studying the provided data, we found out that majority of the people are have the age group xx-24. There were 4 different age groups but overall looked like when we train the data it gets skewed towards the young people age group. Two solutions came up to our mind were training the model with few more datasets or consider less age groups. we was not able to find some similar data online to train our model, so decided to take less groups we decided to work with two younger age groups on local machine i.e xx-24 and 25-34. This helped me to improve our accuracy to 0.69 on the local machine but was not sure if our approach was correct.

The next step was to apply this model to virtual machine. we did the prediction of the data according to their age group whether the user is among the provided 4 age groups. First we wanted to just use all 4 age groups and see results. Then on the virtual machine each user's '.xml' file was produced where all the other information beside age group and gender was hard-coded but the age group prediction was made using the Logistic Regression model. This came out to give us a 0.63 accuracy on virtual machine as the non-administrative user, 'ituser'. We have not had the chance to deploy this model of age prediction on our virtual machine under instructor's evaluation, but it had good local results.

3.2 Relations

For using relation dataset as our source, we have tried several machine learning methods, such as User-Like matrix, User-Page-User matrix, ensemble and k Nearest Neighbors (kNN). The most successful approaches are the User-Like Matrix and the voting for majority method.

3.2.1 User-Like Matrix. In this method, we use each 'like_id' as if they were words in texts, this way, every user will their own transcripts on what they have liked. This the most simple to understand and easy to implement method of approach. We first do a bit of data processing; we merged and joined both sets (training and testing) of datasets (profile.csv and relation.csv) into matrices of 'userid' and 'like_id'. Then we used the combine data to apply the count vectorizer as well as different machine learning models to obtain the better results in Table 1, Section 4.2.

We have first used this approach to predict gender first, while we have seen a good result, we implement it on age prediction as well.

3.2.2 User-Page-User Matrix. The purpose of this approach is to treat each 'like_id' as if they were words in texts, and find out what kind of users would like them, such male vs. female or what age groups would be attracted to that content.

First, we merged and joined the training profile.csv and relation.csv datasets into matrices of 'like_id' and a list of 'userid' that have liked that page/content. Then, we used it to calculate the average gender, age, personality scores of each page into a Page-Score matrix. After that, from the testing dataset, we calculate the average gender, age, and personality scores based on previous calculated Page-Score matrix for each testing users. By the end, we round up or down their scores to group them into their most related predictions. We have tried trimming all the pages with less than 5 likes (not enough information to yield a reasonable ratio) or more than 800 likes (too much information such that everyone likes this page, which yielded a non-useful ratio).

However, our results (Table 2) were not as great as the results from the user-like matrix method; only gender predictions have barely passed the baselines. With closer observations, we have learnt that the training dataset is very unbalanced in the way that in genders and age groups.

To illustrate, about two-thirds of the users in the training dateset are females and most of them are young people and under 24 years old. This leads to the classifiers used more likely to favour the 'female' gender group or the 'xx-24' age group. Results are shown in Table 2, Section 4.2.

3.2.3 Voting for Majority. This method is a small trial ensemble we did, before we could move on to an overall ensemble with all three sources. As texts and relations sources have gathered good results earlier then images sources, we have took the successful models we have so far from these two and play around with with different combination to obtain the best results. This approach uses different models to collect a list of predictions in genders and age groups, for each users, then take the majority vote in the list as the final prediction. During this process, we put equal weights on every results in the prediction lists.

After we have tried different variations of models, we found out that to best predict genders, we used our logistic regression model from relations and the logistic regression model from texts. Because there are only two (even number) results in the prediction lists and since the training dataset suggested that more females are using the social media than males, the final prediction would only be 'male' if both models yielded 'male', otherwise, the users would be categorized as 'females'.

To best predict the users' age groups, we used our Naive Bayes model from relations, logistic regression model from relations and the logistic regression model from texts. This time, we have an odd number of results in the prediction lists, we used the 'most_common()' function in Python3 to select on the majority votes and decided on the final predictions.

Table 3, Section 4.2 has shown our results ran under instructor's evaluations on our team virtual machine.

3.2.4 Other approaches. Another method that we have dabbled with is using K Nearest Neighbor algorithm. The first step we did is to obtain the same user-like matrix from the training dataset we used before in Section 3.2.1. Then, for each testing users, we predicted them based on their k nearest neighbors from the training dataset.

In this process, we have only had the time to try the traditional unweighted k Nearest Neighbor algorithm. However, because the initial trials we did using this method have yield very unsuccessful accuracy. One of the possible reasons that lead to this result could be that the training dataset was very unbalanced as we have explain above in Section 3.2.2. We also did not have sufficient time to put into this method, to dig in deeper into its problems.

3.3 Images

In the part where we need to use images as resource to predict users' gender, age and five personality, the recommendations of methods (or resources) that we should use from professor is convolutional neural network algorithm (which is a deep learning algorithm that very good at image recognition) and opencv (which is a library for image processing). We use this two methods in a comprehensive way to build our own model for users' prediction, more specific, we were focused on the gender prediction of users.

3.3.1 Gender Predictions. After skimmed through all the images in training data set, we realized that there is a good amount of noisy data inside of the 9500 images, such as some users use comic as portrait, some users use significant other as their portrait and some portraits have more than one person, etc. Thus, the process of filtering unpredictable images from data set becomes significant for the prediction accuracy. Therefore, we decided to use opencv as our first phrase for image processing.

Inside opencv library, we used haarcascade_frontalface_alt_tree.xml (the following abbreviated alt tree file), haarcascade_frontalface_alt2.xml (the following abbreviated alt2 file), haarcascade_frontalface_alt.xml (the following abbreviated alt file), haarcascade_frontalface_default.xml (the following abbreviated default file) and haarcascade_eye.xml (the following abbreviated eye file) to filter images. In the process of applying all 4 haarcascade frontal face xml files, we knew that the alt tree file has the most accuracy for multiple faces recognition and the default file has the lowest accuracy, so the alt tree file isolated approximately half amount of images out of the training data set but the default file can only isolated couple hundreds images out of the training data set. On top of that, we found that there are different preferences for isolated images among alt, alt 2 and default files. More, we realized that opencv can only recognized one face out of multiple adjacent faces inside the image, that causes some multi-faces images still remain in the data set.

Although we know that opencv is not perfect to isolate multiple faces image and comic image from 9500 portraits, however, it still can reduce certain amount of noisy data in some ways. Our strategy is to have two layers of filtering images, the first layer is to create an array that contained all 4 haarcascade frontal face xml files, then use a loop to go through this array in order to have all recognizable images, which isolated out 116 images from 9500 images. In the second layer, we used eye file to filter the remaining images from the first layer because we knew eye file can isolate out comic images effectively. The final amount of remaining images is 8053 after two layers isolation, which is 85% of 9500 images. We thought that the remaining images amount is still large. The last step in this phrase is to re-size all remaining images to 100x100 dimension, we thought it is helpful for our training model.

The second phrase for the entire process is divided the remaining images data set into two groups(folders) - male and female, this step is a preparation for label later in our code.

The third phrase is our training model. We use the convolutional neural network algorithm to train our pre-processed images data. In our model, we used 3 convolutional layers, 3 pooling layers, 1 dropout layer, 1 flatten layer and 2 dense layers. CNN works by moving a filter across the input image, in our model, we use 3 x 3 filter(we called it feature maps too) for input layer and hidden layers. We also shuffled the training images and use the dropout layer to avoid over-fitting.

In the process to finalize our training model, we used tensorboard to see what combination of layers could have the best result, so we created three arrays and used few loops to accomplish the task, we tested many possible ways, you can see the code in training_model_with_auto_tensorboard python file. However, it would be able to run on virtual machine because we use it internally on our local computer.

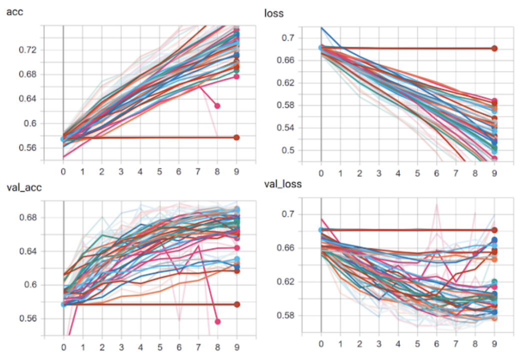

Figure 1: Results from tensorboard

Based on the result we got from tensorboard (we focused on the val_loss and val_acc), it told us that the 3 conv layers with 64 nodes and 2 dense layers that after 6-7 epochs can perform a best prediction for validation data. So we chose this as our base model in the early phrase. Nevertheless, we found that 3 convolutional layers (128-64-64) and dense layers (64-1) with 10 epochs works out better when we used the pre-processed image data set.

The final phrase we have done was save the training result as a model in order to use it in the prediction of test data and applied the same opencv methods for image isolation of test data. For those multi-face images and comic images, we depended on the prediction from relation because the prediction from relation got the highest gender accuracy among three resources.

The last thing we want to mention is we also tried to use images data augmentation in our code but the runtime was super wrong due to the equipment limitation, therefore we did not pursuit that method.

4 RESULTS

For our experiments, we use data obtained from the various processing steps described in Section 3 above. We build models on different regression algorithms like Linear Regression, Ridge, Lasso, Random Forest Regression and Support Vector Regression (SVR). To obtain better results, outliers are removed from the data. We consider the different standard deviations thresholds in removing outliers to see which gives the better results. We split the data as training set and test set with 80% of data as training data and 20% as test data.

4.1 Texts

4.1.1 Gender Predictions. Using the Naive Bayes Model, we have a local 0.70 accuracy for gender prediction.

4.1.2 Age Predictions. Using the Naive Bayes Model, we have a local 0.59 accuracy for gender prediction. Unfortunately, we did not beat the baseline, but it is the exact percentage as the baseline.

We have studied the data provided closely and figured out that most of the people belong to xx-24 age group, as we could say that mostly all the young people use Facebook. Therefore, we applied a different approach for age prediction. While training the data we just took first 2 age groups into account and ignored rest of them this would just give better results on the data set for our project but might have to count for real world data.

After applying this new method, and with the new model of logistic regression, to predict the age of the user using text as an source. This helped us to improve our accuracy to 0.63 in our local machine.

4.2 Relations

Models	Genders	Age Groups
Naive Bayes	0.81	0.65
Logistic Regression	0.82	0.66

Table 1: User-like matrix results running from local computer

Models	Genders	Age Groups
Naive Bayes	0.61	0.52
Logistic Regression	0.62	0.53

Table 2: User-page-user matrix results running from local computer

4.3 Images

4.3.1 Gender Predictions. Locally, we have obtained a 0.5496 in val_loss and 0.7033 in val_acc using solely our image training model without any pre-processed images dataset.

After that, we have improved our model by using pre-processed images dataset. Internally, we have 0.4669 in val_loss and 0.7809 in val_acc with the same training model and 10 epochs, which the result is increased a lot when comparing the result we got earlier internally.

Models	Genders	Age Groups
Baselines	0.59	0.59
"Majority"	0.78	0.67

Table 3: Voting for majority method results running from instructor's formal evaluations

However, the best result we could get from gender prediction under instructor's evaluations on the virtual machine is 0.69 by using just images as resource, there was 9% difference and it is weird. We doubted that there was something wrong on how we process the images in test data side, however, we were out of time to fix that problem.

5 CONCLUSION AND FUTURE WORK

For text as a source, there was an error that kept recurring while the instructor runs evaluations for the age prediction mode on our virtual machine. Unfortunately, due to time schedule to this school quarter, we have left with very limited evaluations on the script and we did not get the chance to do so. Without any questions, for our next step or our future work, this would be the very first step to be taken into consideration while making some progress for text as a source.

For relations as the source, there a few things we could pursuit if we had longer time with this project. First of all, for the user-page-user approach, we could do better job with trimming pages. We could eliminate pages that have equally balanced ratios (e.g. half people who liked a page are females and the other half are males.) We would want pages that are with greatest differences and are really give information on which groups specifically liked them. Hence, with more experiment, we can give better weights to those pages, such that it would produce a such better result.

With the kNN method, we should also start with weighting the distance according to the information gain and use the weighted k Nearest Neighbor algorithm on the relation dataset. We could also drop pages with small number of likes, or easily apply kNN based on these low occurrence pages. These are some of the possibilities for future experimentation.

For the images as the source, we have not figure out the reason why the gender prediction accuracy dropped that much on virtual machine when comparing the result internally. We have tried to debug it but we have not find it out yet. We also want to modify our model to use Stochastic Gradient Descent as the optimizer to explore whether if the gender prediction accuracy could get any better in the future, after we upgrade our testing equipment.

Other than genders and age groups prediction, we have also give a try with the LIWC.csv file to predict users' OCEAN scores. We used a basic count vectorizer and the logistic regression model. The results were about the same as the baselines. We were happy that they were not too bad and we would wish to explore more on predicting these personality traits scores.

Overall, our best results are obtained from using relations as the source in gender prediction. It is associated with the prediction by texts, similar to the best result from images in gender prediction is depended on the prediction by relations partially. Unfortunately, we could not complete the majority voting ensemble method with all three sources. Even though the combination between texts and relations are already a very good start for our ensemble prediction, an overall ensemble would also be a possible next step for us as well.

REFERENCES

[1] PAN, https://pan.webis.de/.

[2] TCSS 455 Machine Learning - User Profiling in Social Media, https://canvas.uw.edu/courses/1301147/files/folder/project?preview=55530146