A Comprehensive Pipeline for Hotel Recommendation System

J. Chen¹, Z. Gao∗¹

¹Department of Computer Science, VU University Amsterdam
vu.gaozhy@gmail.com

Abstract. This paper addresses a comprehensive pipeline to build a hotel recommendation system with the raw data collected by Apps in users’ smartphone. The pipeline mainly consists of pre-process of the raw data and training prediction models. We use two methods, Support Vector Machine (SVM) and Recurrent Neural Network (RNN). The results show that two methods achieved a reasonable accuracy with the pre-process of the raw data. Therefore, we conclude that this paper provides a comprehensive pipeline, in which a hotel recommendation system was successful build from the raw data to specific applications.

1 Introduction

In this paper, we describe a comprehensive pipeline with two methods to predict the mood of the user on the next day based on the data we obtained from the users on the days before. Moreover, we achieve the hotel recommendation for the user based on their mood. The methods are compared with the benchmark that simply predict the mood on the next day by assuming it is the same as the previous day. First, we use the model of Support Vector Machine (SVM) to predict the mood of user. The second method we used in this assignment is the Recurrent Neural Network (RNN). Both of them achieved the prediction in a reasonable accuracy.

The Pre-process of the dataset is a very important step in data mining. Usually, the Pre-process is closely related to the prediction. To be clear description, this document describe the Pre-process of dataset in Section 2. The experiments are implemented in R code based on the libraries like e1071 (SVM), RNN, etc.

2 Pre-process the Raw Data

2.1 Data Analysis

1. Reading and Understanding Data

In this section we use R code to process the dataset due to the plenty of support library in data mining. First, it is necessary to understanding what are the meaning of the variables and the value of the dataset before process the data. The dataset shows the variables and the corresponding values of the users from the smart phone. The mood of the users is related to the variables on the last days. However, the data of some variables are not related to the mood of users, or not unusable due to damage and /or insufficiency. This is the objective in this section, that we aim to pre-process the dataset from an original to the wrapped dataset that can be fed to the predictive model.

2. Pre-process the Dataset

In this section, we present the process about how we pre-process the data. First, to make the data structure clearer, we organize the dataset as the table 1 that the value is grouped by id, time, and variables. In addition, we can analysis the mood of the user in a day like the Figure 1 for the mood dynamic of the user. In such way, we know how is the mood dynamics in a day, which is good to predict the mood of the user in the next day. Therefore we processed the dataset to the new structure as shown in Fig.3. To build the predictive model, we need to summarise the value of the variables in days that can be right format to input the classifiers. We therefore average the value of variables in days.
The mood of an user in a day. The mood of the user keep in a stable average value in [7,8].

The predictive model procedure.

Fig. 3: The snapshot of the data structure that the value is grouped by id, time, and variables. The data in red rectangle are unusable due to too NA.

However, in figure 3, we can see some variables of dataset we obtained have few data. We think they are unusable data, and remove them. Although the dataset is much more tidy, it is still not good enough to be the training data and test data. Therefore we remove the data in some days that only have a little value of variables. To here, the dataset are usable for training and testing as shown in Fig. 4.

To pre-process the dataset to be usable in data mining, we face many challenges and problems to the original dataset like missing value, outliers, etc [1]. The pre-process is an essential and important step for data mining due to a variety of possible defect in the original dataset.

Here we show many examples that are part of the techniques we used in our experiments. Missing value in original dataset is a common problem that we have to solve in data mining. First, we illustrate how we process the problem of missing value in the original dataset.

2.2 The set-up of feature

In Fig. 4, we choose some variables as the usable feature with enough samples. The data in Fig. 4 have the full values in the variables in different dates and ids. They are tidy data that can be fed to the model for training and testing from the data format and information.

In other hand, we divide the dataset into two parts with 10% and 90% rate as the training sample and testing sample separately. To build the predictive model as shown in Fig. 2, we aggregate the history to create attributes that can be used in the machine learning approach like the SVM [2] and RNN we use in this document. We use the average mood during the last five days as a predictor. This is clearly present in Fig. 5 to create the new feature that can be used in the classifiers.
Table 1: The re-structure of dataset.

| id     | time         | mood | variables |
|--------|--------------|------|-----------|
| AS14.01 | 2014-02-26   | 6.25 |           |
| AS14.01 | 2014-02-27   | 6.33 |           |
| AS14.01 | 2014-03-21   | 6.2  |           |
| AS14.01 | 2014-03-22   | 6.4  |           |
| AS14.01 | 2014-03-23   | 6.8  |           |

Fig. 4: The snapshot of the usable data structure. The value is grouped by id, time, and variables. There are no NA and duplicated id in different variables.

Fig. 5: The example of temporal abstraction. We take the average of mood value in five days.
2.3 Rationale

For the rationale of choice of the final attributes, in this assignment, we mainly consider the quality and quantity of dataset. We have to filter damaged data that probable to train an incorrect predicted model or decrease the accuracy of the prediction. Therefore, we remove the data in the day that many variables have missing value and the variable with outliers.

3 Learn prediction models

In this section, we described two predictive models and the benchmark. But we are not focus on the details of the models because we use the standard R code library for SVM and RNN.

3.1 Model Variant 1

First, we adapted the Support Vector Machine (SVM) as the predictive model. In R programming, a variety of libraries can be used to implement SVM, we used the library e1071 due to its feature of easy-to-use. The main parameters setting of SVM is shown in Table 2.

| parameters | scale | type      | kernel | degree | gamma coef0 | cost | class.weights | epsilon |
|------------|-------|-----------|--------|--------|--------------|------|---------------|---------|
| setting    | 1     | C-classification | linear | 3      | 1            | 0    | 1             | 0.1     |

Table 2: The main parameters setting of SVM.

Therefore, we just need to train and test the sample after the pre-process section. We used the variables in section 2 as the input of SVM model and the value of mood is the output of SVM model. The 90% sample was used to train the SVM model. The rest 10% sample was used to test the accuracy of trained SVM model.

We output the accuracy of trained SVM model by predicting the training sample. And then, The accuracy was verified again by predicting the testing sample. The table 3 shows the results of predictive mood of the user on the next day that testing on the training sample. In table 3, the 568 samples are used to train the SVM predictive model. The value of mood from 5 to 8 are predicted from 3 to 9. The 467 samples are correct predicted. The trained SVM model have accuracy = 467/568 = 82.2%.

| results_train | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---------------|---|---|---|---|---|---|---|
| 5             | 0 | 1 | 4 | 1 | 0 | 0 | 0 |
| 6             | 1 | 2 | 9 | 71| 11| 4 | 0 |
| 7             | 0 | 0 | 0 | 23| 31| 31| 2 |
| 8             | 0 | 0 | 0 | 0 | 13| 79| 0 |

Table 3: The results of training. The 568 samples are used to train the SVM predictive model. The value of mood from 5 to 8 are predicted from 3 to 9. The 467 samples are correct predicted. The statistical results in Fig. 6.

Furthermore, we verified the accuracy of SVM predictive model by predicting the test sample. We have divide 100 sample as the testing sample in section 2. We have the result as shown in Table 4 and the statistical results in Fig. 7. The 81 samples were correctly predicted.

Last, we test the accuracy of benchmark that assuming the mood of user is the same as the previous day, which is 62.3%. Therefore, we have the comparison as shown in Table 5.

3.2 Model Variant 2

For this variant of the model, we incorporate a Recurrent Neural Network (RNN) to exploit the temporal characteristics of the dataset. To do so, we first pre-process the data somewhat. For this
pre-processing we first replaced all the unavailable values, that is the values corresponding to ‘NA’, by their values of the previous data-point. In such a way, we can use more data-point and do not have to discard any data points. Besides that, it seems reasonable to equate these values to their previously measured values since all the variables are measured several times a day, and it seems plausible that the values of these variables do not change substantially from one data-point to the next.

Now that we have a full dataset with no missing values, we can aggregate the data over the days. This allows us to obtain averages of all the days for each variable, which is needed to produce a prediction of the average mood the next day. At the same time, all the days that the mood variable is measured and delete the days in our dataset for which the mood is not observed for each individual separately. By doing this for each individual separately, we avoid throwing away data that we could in fact use for certain individuals. So if, for instance, the mood is only measured for an individual 1 at 10 dates, but for some other individual 2 on 15 dates, we avoid throwing away 5 dates for individual 2. As a next step, we then find for which days, where at least the mood has been measured, the most variables have been recorded and discard the rest of the days in our dataset. This is again done for each individual separately, which results in the following number of observations.

As a final step for preparing to fit a RNN, we convert all variables to the $[0,1]$ interval, which is necessary for the RNN to converge (faster). In the end, we scale back our predictions to their original scale such that we obtain predictions for the mood that we actually observe.

For every individual we then train and test a RNN, where we eventually ended with a learning rate of 7%, 4 hidden layers in the network, 10,000 iterations, the logistic sigmoid and the stochastic gradient descent method as updating rule. For testing the individual RNN’s, we used 30% of the available data (rounded to the nearest integer) and the other 70% (rounded to the nearest integer) for training the data. As an example, we present the results of this training and testing phase.
Table 5: The comparison of testing based on training sample and testing sample, and the benchmark

| prediction result_train | result_test | benchmark |
|-------------------------|-------------|-----------|
| accuracy                | 0.822       | 0.810     | 0.623     |

Table 6: Number of observations after pre-processing the data for individual AS14.08 below. Note that for each individual, the random number generator in the training phase was initialized by `set.seed(2204)`.

(a) Error for training set of AS14.08

(b) Predictions for test set of AS14.08

Fig. 8: Training phase for AS14.08
From results in Figure 8, the errors made in classifying the data decrease rather steep as the number of iterations progress. From the corresponding prediction plot, we see what the actual values of the mood were in the test set as opposed to the predicted values from the trained RNN. We might be worried from the error plot that we are overfitting the data in the RNN since the errors become so small, but we can see that the RNN seems to reasonably predict the mood for the following day from the prediction plot. This means our RNN is not overfitting in this case and that it can reasonably predict the mood for the following day for unknown cases. If we train our network using the entire dataset, we can also see that we adequately capture the mood of the following day for the known cases.

![Actual values vs predictions](image)

**Fig. 10: Predictions for entire dataset of AS14.08**

The results of predictions in Figure 10 show that we actually capture the mood of the following day with rather high accuracy. As expected, we thus obtain qualitatively the same pattern for the errors made in classifying the data as for the training phase earlier arises when we use the entire dataset, explains why we are able to predict the mood of the following day quite precisely. Moreover, this pattern for both the predictions and the errors is consistent for all the individuals. To provide a selection of our results, we show the prediction plots for 3 individuals below, namely for AS14.08, AS14.16 and AS14.24. From these plots we indeed see that our predictions match the observed values rather closely and seem pretty accurate. This is confirmed when we inspect the RMSE of the predictions for each individual. These RMSE’s are given in the following table. We

| ID    | RMSE | ID    | RMSE |
|-------|------|-------|------|
| AS14_01| 0.4013390 | AS14_19| 0.4711904 |
| AS14_02| 0.2142030 | AS14_20| 0.3762178 |
| AS14_03| 0.4265163 | AS14_23| 0.2910442 |
| AS14_04| 1.0045396 | AS14_24| 0.2965332 |
| AS14_05| 0.2778267 | AS14_25| 0.3899919 |
| AS14_06| 0.2778267 | AS14_26| 0.2770717 |
| AS14_07| 0.5791765 | AS14_27| 0.3803674 |
| AS14_08| 0.5235971 | AS14_28| 0.2794391 |
| AS14_09| 0.3836072 | AS14_29| 0.5301942 |
| AS14_10| 0.4786648 | AS14_30| 0.2453128 |
| AS14_11| 0.2647090 | AS14_31| 0.3185494 |
| AS14_12| 0.2972429 | AS14_32| 0.2421542 |
| AS14_13| 1.0184591 | AS14_33| 0.3605297 |
| AS14_14| 0.4683970 |

**Table 7: RMSE of RNN approach**
see that these RMSE’s are all pretty close to zero for each individual. All things considered, we thus see that the predicted values match the actual values rather closely, for each individual. The RNN method thus seems to adequately incorporate the temporal aspects of the dataset at hand on an individual level.

3.3 Model Variant 3

In this model variant, we simply predict that the average mood on the next day is the same as on this day. In the prediction plots below we can see these actual and predicted values for 3 individuals, namely for AS14.08, AS14.16 and AS14.24. We can see that this naive approach of simply predicting that the average mood will stay constant (the same as the previous day) does not produce as nice results as those of the RNN’s. The following table shows the corresponding RMSE for each individual when adopting this naive approach.

| ID    | RMSE  | ID    | RMSE  |
|-------|-------|-------|-------|
| AS14.01 | 3.802594 | AS14.19 | 4.642377 |
| AS14.02 | 6.134013 | AS14.20 | 3.289039 |
| AS14.03 | 2.906315 | AS14.23 | 3.544714 |
| AS14.05 | 5.314184 | AS14.24 | 5.064912 |
| AS14.06 | 4.985479 | AS14.25 | 3.896437 |
| AS14.07 | 10.497090 | AS14.26 | 6.547519 |
| AS14.08 | 4.977672 | AS14.27 | 5.096676 |
| AS14.09 | 5.282255 | AS14.28 | 5.410381 |
| AS14.12 | 4.506662 | AS14.29 | 4.377468 |
| AS14.13 | 7.353911 | AS14.30 | 2.966854 |
| AS14.14 | 4.541047 | AS14.31 | 3.009430 |
| AS14.15 | 3.139621 | AS14.32 | 4.637708 |
| AS14.16 | 4.735328 | AS14.33 | 7.052462 |
| AS14.17 | 3.596140 |       |       |

Table 8: RMSE of naive approach

From this table we also see that the RNN’s actually perform much better than the naive approach. All things considered, the naive approach does not seem correct to adopt and can be considered as a ‘clueless’ method, that is if we had no idea how to approach the problem then this would be the standard ‘worst case scenario’ for producing predictions. The naive approach can therefore indeed be considered as a benchmark model.
Fig. 13: Actual values vs predictions for each individual (1)
4 Conclusion

Hotel recommendation system is a popular research field. This paper provides a comprehensive pipeline for the researchers to build such a system from the raw data to specific application. Although the results show that the two methods achieve a successful prediction system, they are only the basic approaches in machine learning. Many approaches are interesting to further exploration. For instance, evolutionary approaches have been applied in many areas [3]. Neuroevolution has been applied in evolving neural network for real-time computer vision [4], evolutionary robotics [5,6,7,8]. In many areas [9], convolution neural networks generally achieve remarkable performance that we aim to apply in this pipeline. Knowledge graph is a popular method that is applied to the many applications [10,11], such as finance, medicine, biology, Question—Answering, Storing Information of Research, in particular recommendation system. Therefore, we will use knowledge graph to design the hotel recommendation system in the future. In addition, the signal compress [12,13,14] is an interesting technology for the pre-process raw data. These approaches are the points we aim to extend for this pipeline.

References

1. Dunren Che, Mejdl Safran, and Zhiyong Peng. From big data to big data mining: challenges, issues, and opportunities. In International conference on database systems for advanced applications, pages 1–15. Springer, 2013.
2. G. Lan, J. Benito-Picazo, D. M. Roijers, E. Domínguez, and A. E. Eiben. Real-time robot vision on low-performance computing hardware. In 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), pages 1959–1965, Nov 2018.
3. Gongjin Lan, Jakub M. Tomczak, Diederik M. Roijers, and A. E. Eiben. Time efficiency in optimization with a bayesian-evolutionary algorithm, 2020.
4. G. Lan, L. de Vries, and S. Wang. Evolving efficient deep neural networks for real-time object recognition. In 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pages 2571–2578, Dec 2019.
5. G. Lan, J. Chen, and A. E. Eiben. Simulated and real-world evolution of predator robots. In 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pages 1974–1981, Dec 2019.
6. Gongjin Lan, Matteo De Carlo, Fuda van Diggelen, Jakub M. Tomczak, Diederik M. Roijers, and A. E. Eiben. Learning directed locomotion in modular robots with evolvable morphologies. ArXiv, abs/2001.07804, 2019.
7. Gongjin Lan, Jiunhan Chen, and AE Eiben. Evolutionary predator-prey robot systems: from simulation to real world. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pages 123–124. ACM, 2019.
8. Gongjin Lan, Milan Jelisavcic, Diederik M. Roijers, Evert Haasdijk, and A. E. Eiben. Directed locomotion for modular robots with evolvable morphologies. In Parallel Problem Solving from Nature – PPSN XV, pages 476–487. Springer, 2018.
9. Guocheng Liu, J. Liang, G. Lan, Q. Hao, and M. Chen. Convolution neural network enhanced binary sensor network for human activity recognition. In 2016 IEEE SENSORS, pages 1–3, Oct 2016.
10. Ting Liu, K. Anton Feenstra, Jaap Heringa, and Zhisheng Huang. Influence of gut microbiota on mental health via neurotransmitters: A review. Journal of Artificial Intelligence for Medical Sciences, 1:1–14, 2020.
11. Ting Liu and Zhisheng Huang. Evidence-based analysis of neurotransmitter modulation by gut microbiota. In International Conference on Health Information Science, pages 238–249. Springer, 2019.
12. G. Lan, X. Hu, and Q. Hao. A bayesian approach for targets localization using binary sensor networks. In 2015 8th International Symposium on Computational Intelligence and Design (ISCID), volume 2, pages 453–456, Dec 2015.
13. G. Lan, J. Liang, G. Liu, and Q. Hao. Development of a smart floor for target localization with bayesian binary sensing. In 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), pages 447–453, March 2017.
14. Gongjin Lan, Ziyun Luo, and Qi Hao. Development of a virtual reality teleconference system using distributed depth sensors. In 2016 2nd IEEE International Conference on Computer and Communications (ICCC), pages 975–978, Oct 2016.