Eavesdropping - Packet Classification

Digital Forensics

A.Y. 2018-2019

Padoan Alberto - ID 1179704
Sansoni Marco - ID 1179411

INTRODUCTION

Nowadays mobile devices are more and more important in our daily life. Differently from about a decade ago, smartphone are not used only for calls and SMS but mainly for application based on the network traffic and, in addition, there are rising a lot of alternatives to bring to the internet-based usage some applications which are not based on the internet protocol (think about calls and messages). For this reason, since mobile devices contain and exchange huge quantity of sensible information, an actual challenge is to ensure the security of such devices.

If the transmission does not provide any type of secrecy and integrity protection service, it is easy for an eavesdropper to capture and read all the data. Fortunately most of the mobile applications are protected by the Secure Socket Layer (SSL) and the more efficient Transport Layer Security (TLS) but, in practice, these protocols do not protect users from any type of illegal access attack. We focus our work - based on the researches of Conti, Spolaor, Mancini and Verde of the Universities of Padova and Roma [1] - on demonstrating that it is possible to detect, by using advanced machine learning algorithms, to which type of operation belongs a sniffed packet. We used a dataset provided by the Math Department of the University of Padova [2]. The dataset contains an huge number of sniffed TCP/IP packets with the related information (source/destination IP, IP ports, incoming/outgoing size, timing etc.). Each packet flow is labelled with the application involved (Facbook, Twitter, Gmail, etc.) and the type of operation performed in each application, which is exactly what we are going to detect with our method.

Packet classification algorithms could be used for many purposes such as:

- a censorship government may use this method to identify dissidents by analysing the operations they are doing;
- it is possible to trace, identify and study the habits of a person and, eventually, his/her relations too;
- during a digital forensic investigation it is possible to detect the actions of a investigated person and use them as a proof.

In our work we decided to use two type of machine learning structures - a random forest and a neural network - in order to evaluate their performances, pros and cons.

Dateset

The dataset, provided by the Math Department of the University of Padova [2], contains 252150 records; each record corresponds to a specific action related to an application (Facebook, Twitter, Gmail, Dropbox, Tumblr, Evernote etc.) and for our work we decided

to perform our analysis on Facebook, Twitter and Evernote applications. Each record of the dataset, in practice, is a flow of packets and it is represented by some features:

- action_start = time in which the action is started by the user;
- flow_number = identifies the flow in the dataset;
- IP_destination = destination address of the packets;
- IP_dest_resolved = resolved IP address of the outgoing packets;
- port_source = port number of the source;
- port_destination = port number of the destination;
- flow_length = number of packets in the flow;
- flow_start = time in which the flow starts;
- packets_length_total = length of both incoming , represented by a plus sign, and outgoing packets which are represented by the minus sign.

Data preprocessing

First of all we divided data according to the type of applications we want to test (Facebook, Twitter, Evernote). Then, in order to have an homogeneous dataset, we created a new category named 'other' which contains all the operations that do not provide a relevant amount of data. We obtain, for each application, a dataset represented by Figures 1, 2, 3 in which is possible to appreciate all the type of operations involved in our work and their amount of available data.

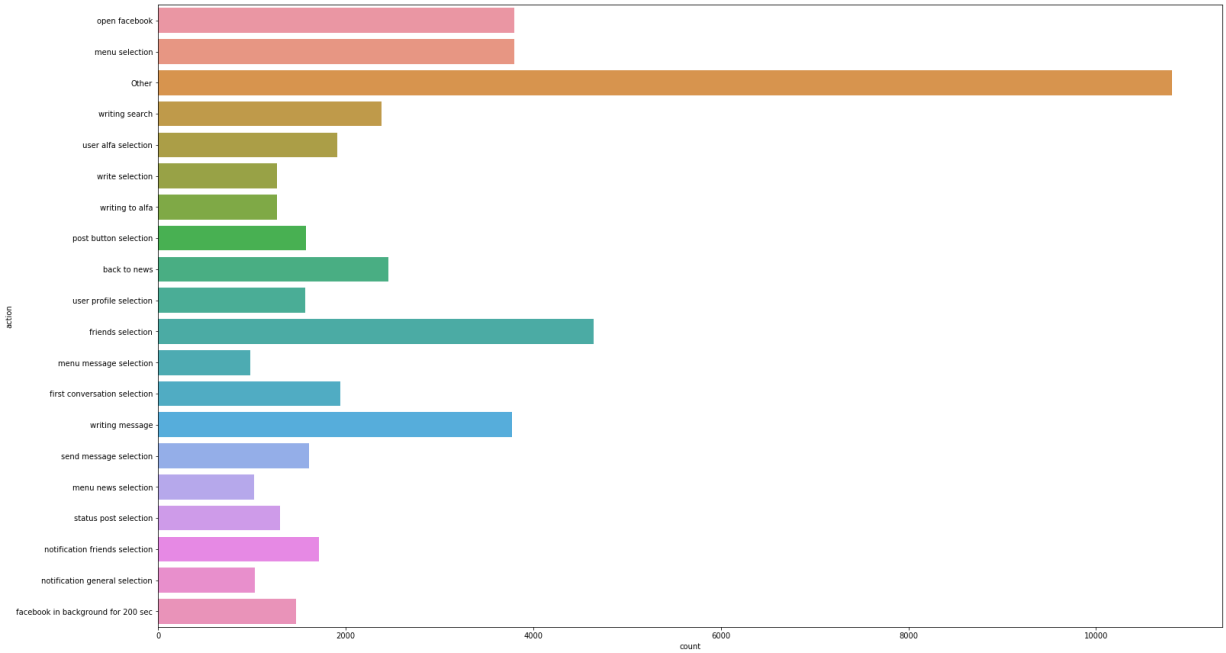

Figure 1: amount of data for Facebook operations

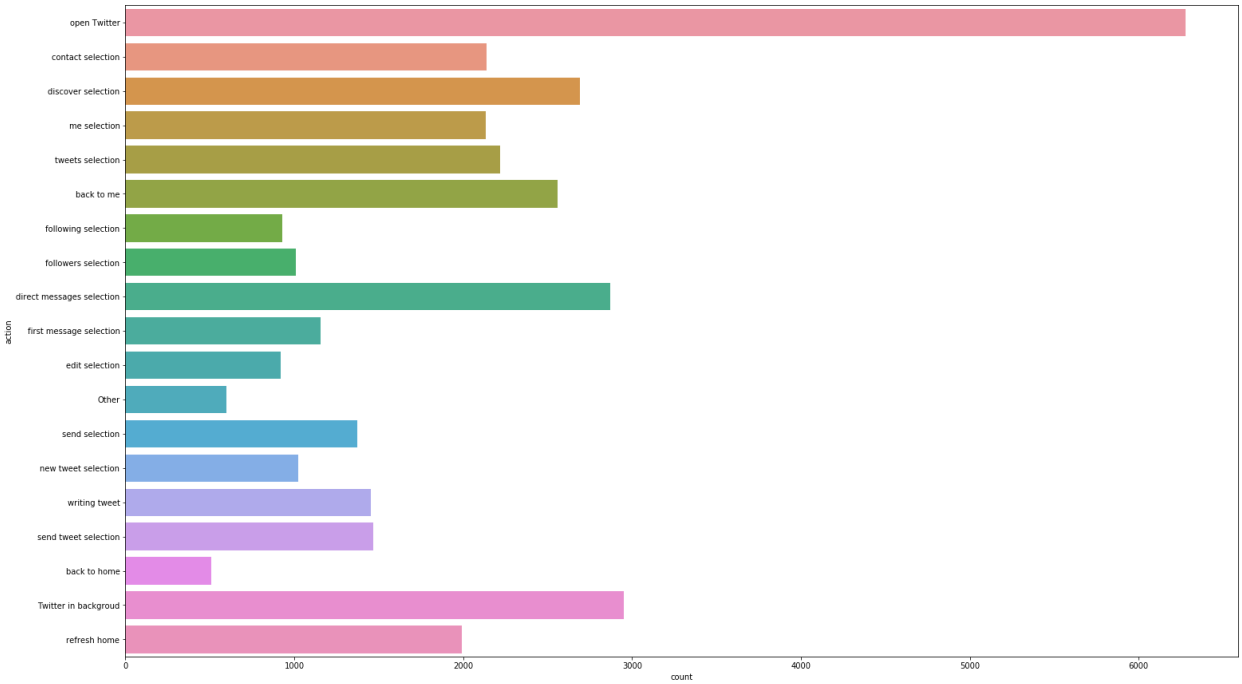

Figure 2: amount of data for Twitter operations.

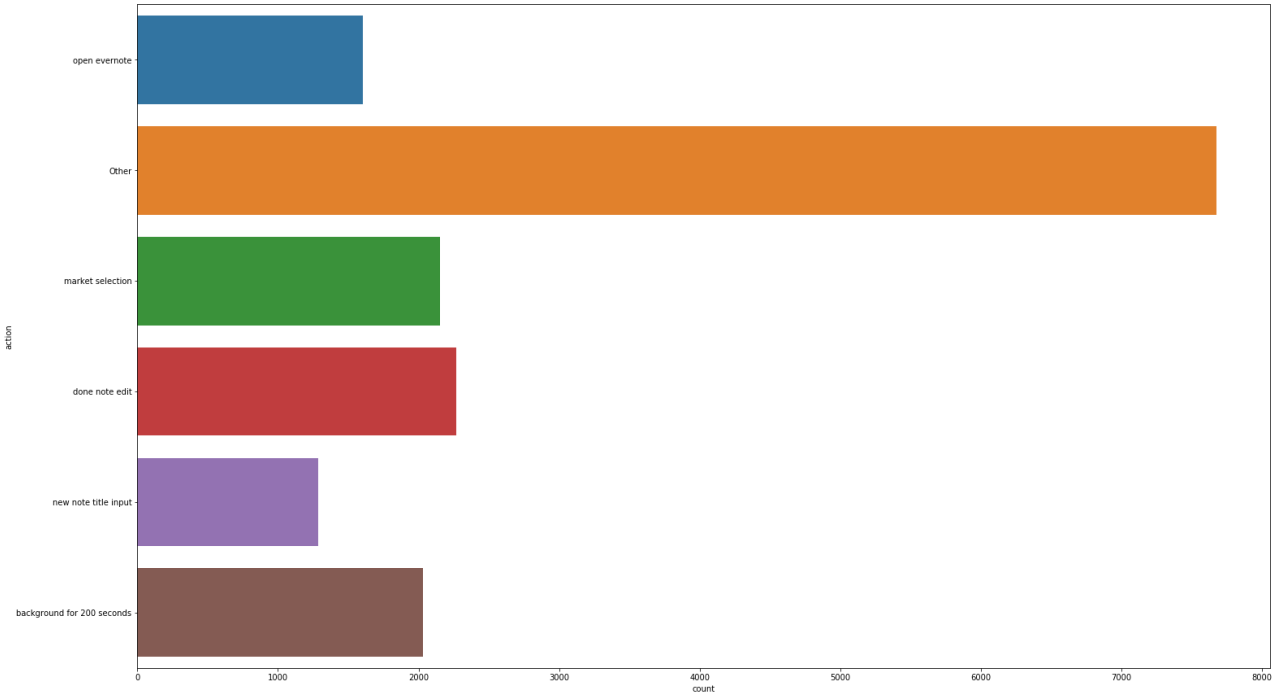

Figure 3: amount of data for Evernote operations.

In order to have a more meaningful and complete dataset, we computed some features related to the available data. For each flow we computed a weighted average of the packets length, in which incoming packets are more relevant (the weight is 0.67) than the outgoing packets (0.33). And, in addiction, we computed the variance of the packets length too. Moreover, since we noticed that the packets that are involved in a single

operations come in groups, we decided to add some features that correlates each flow to the flows that compose a "block", i.e. the operation. Since the preprocessed data are later divided in training and test set, we computed the block features by splitting flows exploiting their properties instead using the label provided in the dataset. Specifically, we stated that a flow belongs to a block if it comes before a time threshold; in other words, if two flows come in a small time interval, they belong to the same block i.e. they are processing the same operation. The threshold we chose is 4.5 seconds since in [1] is demonstrated that 95% of all packets come at most 4.43 seconds after their predecessor. So, for each flow we added the following features:

- Num_flow = number of flow in a block;
- Duration = duration of the block ;
- Shift_from_start = time difference between the actual flow and the first flow in a block;
- Avg_flow_length = (max time in the block - min time in block)/Num_flow;
- Mean_block = the mean combining all the packets length of all the flow in a block;
- Var_block = variance combining all the packets length of all the flow in a block.

Then we removed the features that we did not retain relevant for the study such as the 'flow_start' that we used to compute the timing features related to the block, the 'IP_destination' and 'IP_dest_resolved' that are present only in a few number of flows and do not have a meaningful value related to the operation type, the 'packets_length_total' that we used to compute the mean and the variance of the flow etc..
And so, in addiction to the features previously listed, we used also:

- flow_length = number of packets in a flow;
- mean = weighted mean of the packets length in a flow;
- variance = variance of packets length in a single flow.

Finally data are shuffled and split in training set (70%) and test set (30%).

Method 1 - Random forest

Random forest is a method for classification and regression that is composed by a multitude of decision trees. Each tree produces an output and the final output of the random forest is evaluated by a voting method combining the output of every tree. For our tests we built a random forest with 1000 trees which maximum depth is 30. We have chosen these values since we noticed that, by expanding the forest, there was not a significant improvement in the accuracy. Then, even if it is a classification problem, we use a regressor. It means that the output will be a number equal to the number of the corresponding label. In Figure 4 there is the representation of some nodes of one of the decision tree used for Evernote. It is possible to see that for each leaf there are as many values as the number of possible action to classify and there is a 1 in correspondence of the leaf output. So, when a generic input starting from the root reaches a leaf, the output of that decision tree is exactly the action related to the value 1.0 of that leaf.

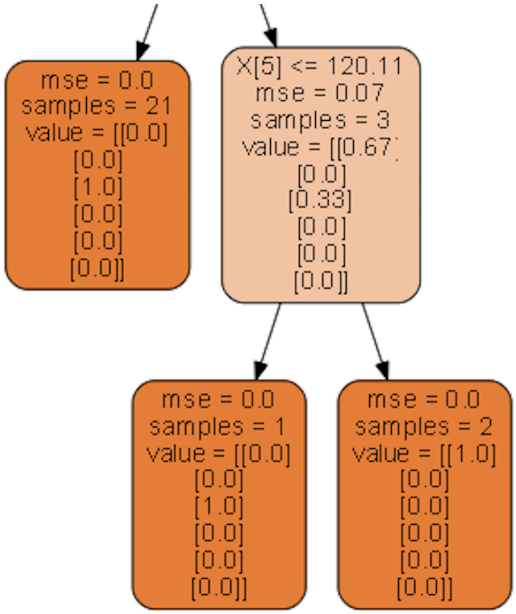

Figure 4: nodes of a decision tree.

Once we trained the model, it is possible to obtain the results of the trained model giving in input the test set. We defined an accuracy function too. We compute each predicted label of the model, that is the label which obtained the higher score by the decision trees. If the predicted label is different from the actual label, it is counted as misclassified. Finally the accuracy is the portion of well-classified labels. In addiction, we display the time required by the algorithm for both prediction and training. The results related to each application are reported in the tables below.

Facebook	
Number of labels	20
Size of training set	35223
Training time	117.49 s
Size of test set	15096
Number of misclassified samples	1859
Test accuracy	87.69
Prediction time	1.83 s

Twitter	
Number of labels	19
Size of training set	25381
Training time	79.99 s
Size of test set	10878
Number of misclassified samples	1256
Test accuracy	88.45
Prediction time	1.08 s

Evernote	
Number of labels	6
Size of training set	11909
Training time	32.05 s
Size of test set	5105
Number of misclassified samples	492
Test accuracy	90.36
Prediction time	0.67 s

Then, in order to give a general vision of the disposition of the misclassified samples, it is provided the plot of the confusion matrix for the Evernote application (Figure5).

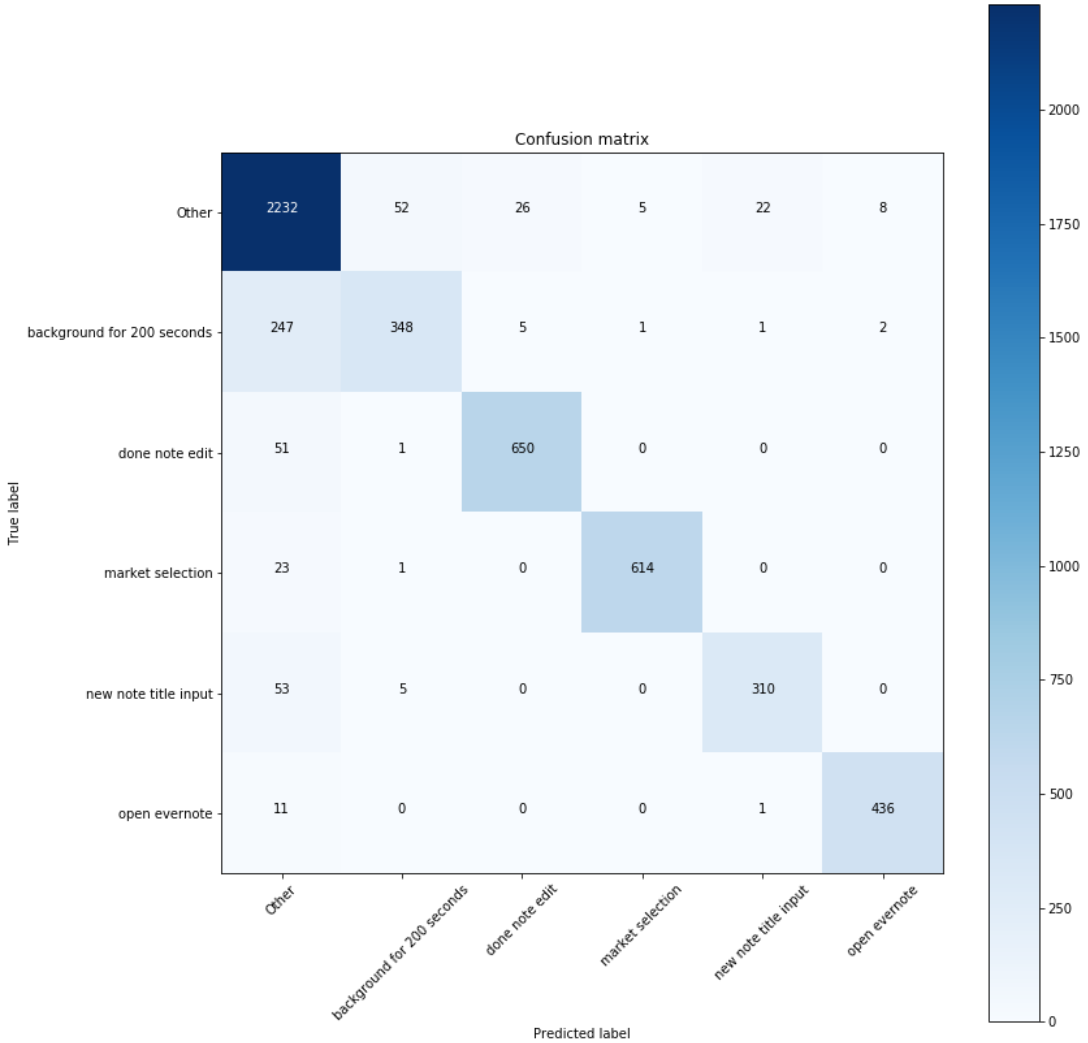

Figure 5: confusion matrix for Evernote.

As is reasonable to expect, most of the errors are due to samples wrongly classified as 'other'. In fact, the 'other' class contains noisy samples since, by construction, originally those samples belonged to a wide range of classes.
Then, in order to obtain a slightly improvement of the performances, we performed a sort of feature selection. Since it is possible to extract the weights that the random forest gives to each feature, we decided to drop less significant features, that are features with a weight less than 0.1. In the following table the features weights are resumed and the less significant weights are highlighted.

Feature	Facebook	Twitter	Evernote
flow_length	0.0236	0.0017	0.0157
num_flow	0.1903	0.1876	0.1071
duration	0.2445	0.2974	0.1622
avg_flow_length	0.1194	0.2158	0.2464
mean_block	0.1339	0.1588	0.1478
var_block	0.1536	0.1012	0.1765
position	0.0032	0.0018	0.0053
shift_from_start	0.0259	0.0173	0.0393
mean	0.0574	0.0056	0.0636
variance	0.0483	0.0127	0.0362

So the model is trained with the reduced set of features, as the prediction with the test set. The results for each application are reported in the tables below and compared with the previous results. It is possible to notice a slight improvement with respect to the previous results.

Facebook	
Previous training time	117.49 s
New training time	69.21 s
Previous prediction time	1.83 s
New prediction time	1.78 s
Previous number of misclassified samples	1859
New number of misclassified samples	1690
Previous test accuracy	87.69
New test accuracy	88.80

Twitter	
Previous training time	79.99 s
New training time	45.05 s
Previous prediction time	1.08 s
New prediction time	1.23 s
Previous number of misclassified samples	1256
New number of misclassified samples	1120
Previous test accuracy	88.45
New test accuracy	89.70

Evernote	
Previous training time	32.05 s
New training time	18.20 s
Previous prediction time	0.67 s
New prediction time	1.49 s
Previous number of misclassified samples	492
New number of misclassified samples	361
Previous test accuracy	90.36
New test accuracy	92.93

Finally, in Figure 6, there is the new confusion matrix for the Evernote app.

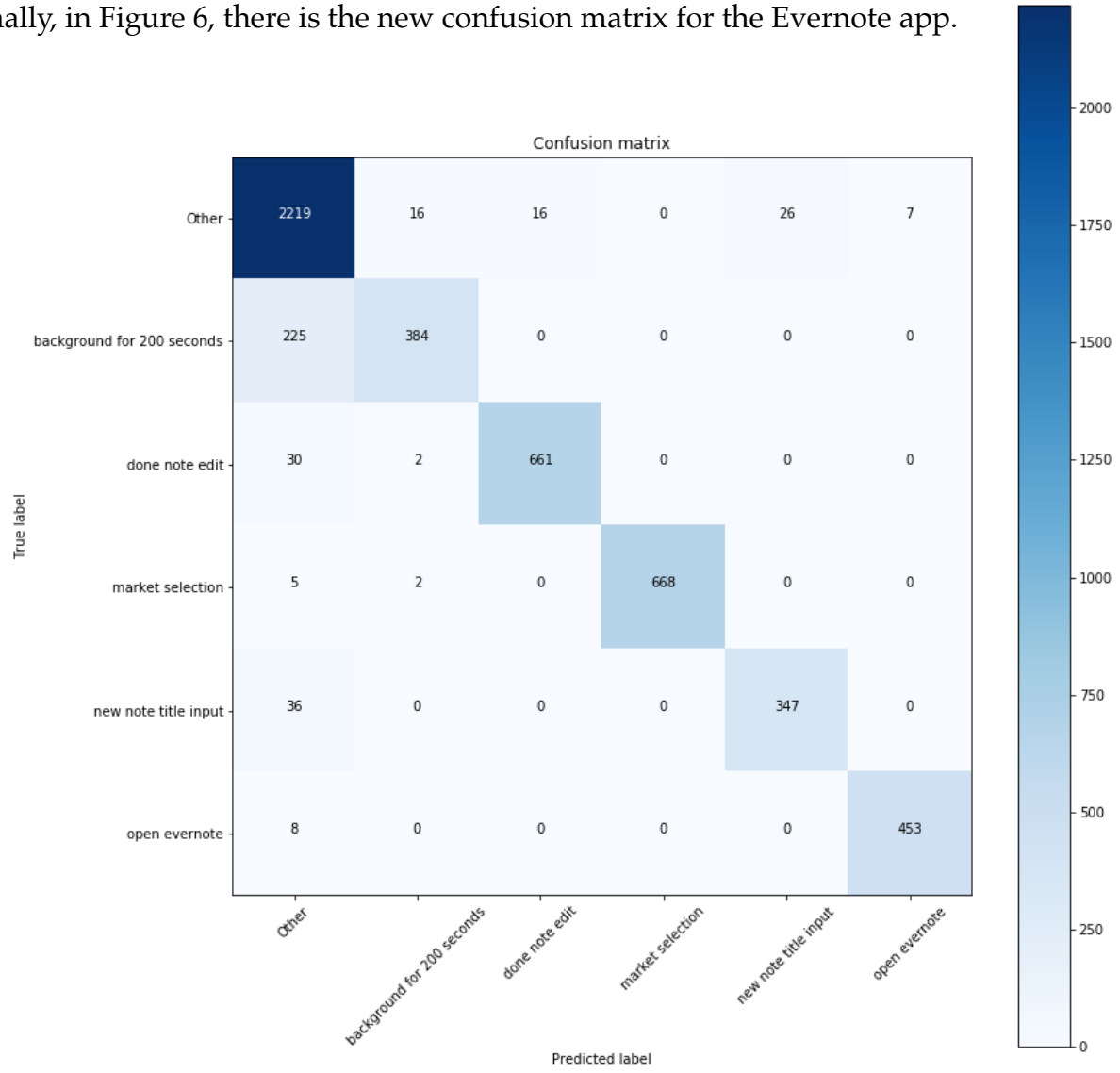

Figure 6: confusion matrix for Evernote app after feature reduction.

Although the improvement is not very significant, by analysing the last confusion matrix, it is evident that after feature reduction the noise due to 'other' class is a bit reduced.

Method 2 - Neural Network

We test our dataset on a neural network too. Although the results are not satisfying as the one obtained with the random forest, we report them for the sake of completeness.
We have built a neural network with 5 hidden layers; each hidden layer is a dense layer with *relu* activation function, since it is an activation function that offers good performances for hidden layers. Instead, the output layer works with the *softmax* activation function, that suits well for the output of a classification problem. Finally, between hidden layers, we performed a dropout which removes the 30% of the nodes in order to prevent overfitting. The representation of the resulting neural network is figured in Figure 7.

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 512)	5632
dense_2 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8256
dropout_3 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 32)	2080
dropout_4 (Dropout)	(None, 32)	0
dense_6 (Dense)	(None, 20)	660

Total params: 180,852
Trainable params: 180,852
Non-trainable params: 0

Figure 7: neural network structure

The training of the neural network is performed on the same dataset used for the random forest without feature reduction, so labels and features are the same as before. The network is trained by using 500 epochs and a batch size of 150: we found this tradeoff between speed (an epoch keeps about 2 seconds to train the network) and efficiency since the training accuracy is slightly better epoch by epoch until it converges to the value that is less or more 75% of accuracy. The performances of the network are compared with the one of the random forest and they are reported in the table below. The results below are only of the facebook app since the other are very similar and not relevant for the comparison of the two machine learning algorithms.

	Neural network	Random forest
Training time	1221.50 s	69.21
Prediction time	0.60	1.78
Test accuracy	78.72	88.80

As we expected the training time of the neural network is sensibly higher, however the prediction time of the neural network is about a third of the one of the random forest. In addiction random forest offers better results on test accuracy than the neural network and makes that data structure the one that suits better for our classification problem.
For sake of completeness we report in Figure 8 the plot of both training loss and training accuracy of the involved neural network.

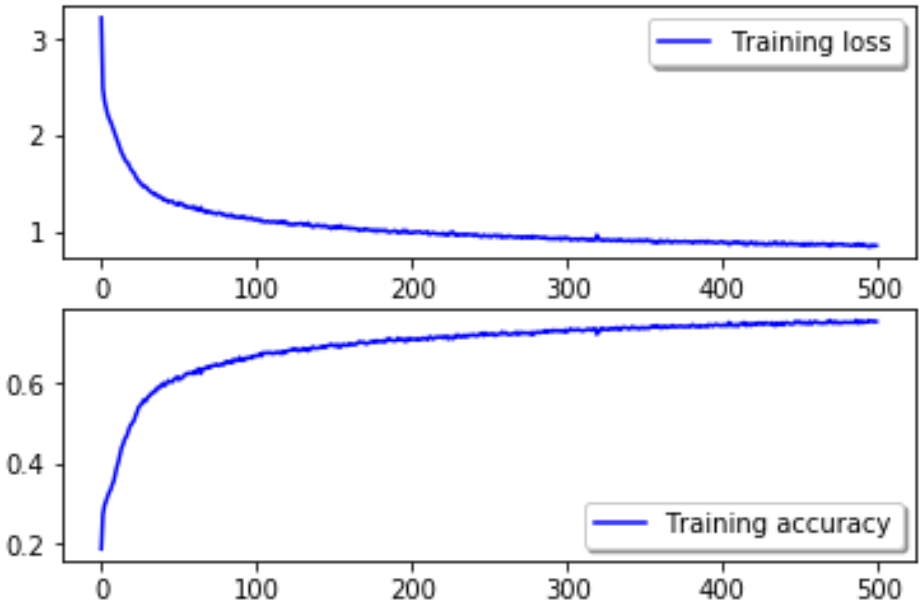

Figure 8: training loss and test accuracy of the neural network

Then, in order to comprehend the classification difference between the two machine learning algorithms, in Figure 9 it is displayed the confusion matrix resulted by running the neural network for the Evernote app.

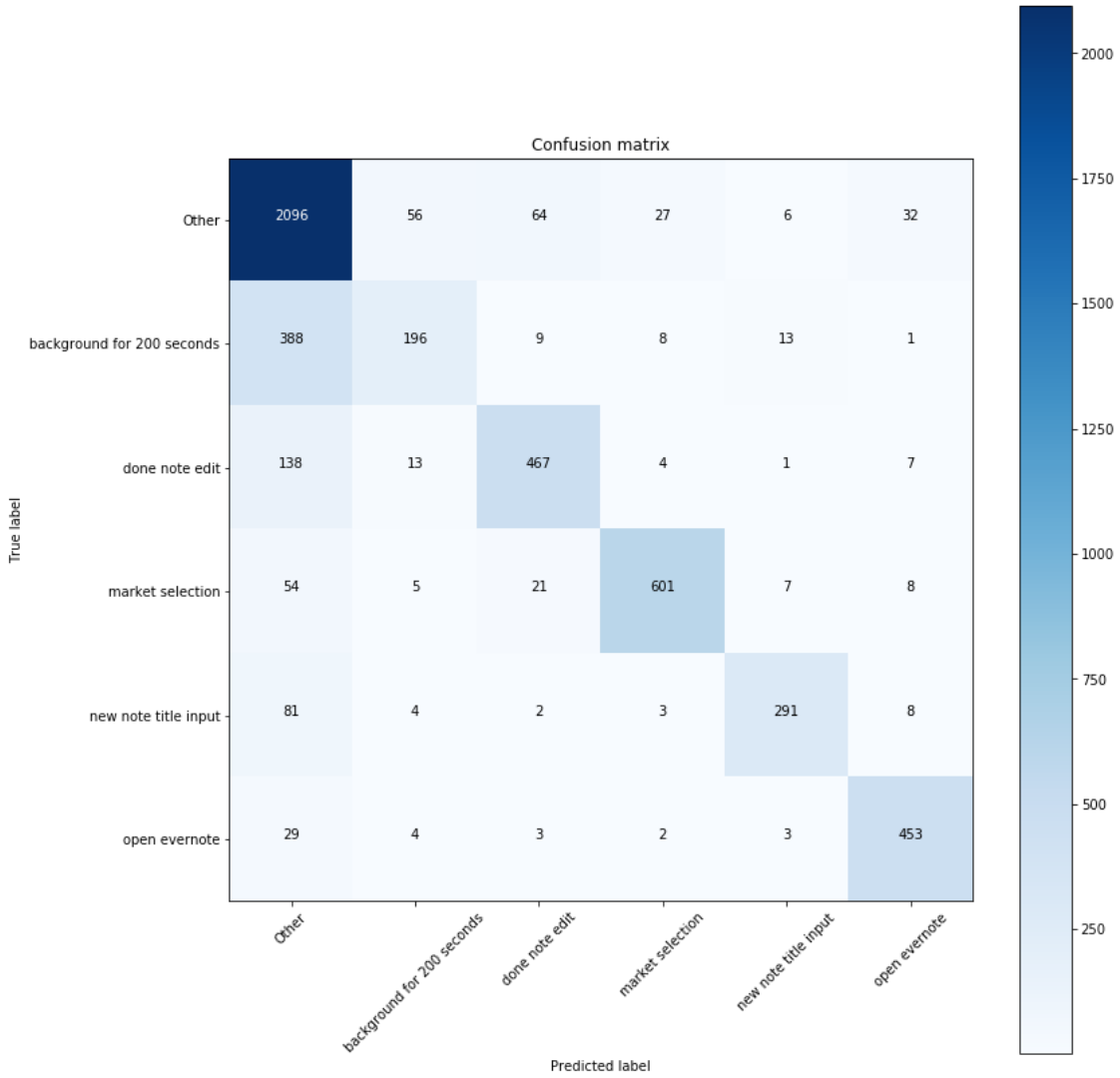

Figure 9: confusion matrix for Evernote app processed with the neural network

As before, although the errors are more frequent, they are always due to the noise introduced by the 'other' class giving a further confirm of this behaviour.

Conclusions

In our tests we performed two type of machine learning algorithms. As we saw in our results the random forest is the one that performs better, and for training speed and for accuracy; in fact the random forest can deal better with the noisy 'other' class and with the high quantity of classes involved in the classification problem. According to this, one possible future improvement could be the removal of the 'other' class, and, if some classes are not interesting for the classification, they could be removed in order to perform the machine learning algorithms with a restricted number of classes: performance could be sensibly improved.
Finally, in order to have a comprehensive view of our results, we report below the tables that resume the performances related to every operation of Facebook, Twitter, and

Evernote. For each operation are provided the number of samples, the misclassified one, and the relative accuracy for the random forest, the random forest with feature reduction, and the neural network.

	Action	Number of sample	Misclassification	Accuracy	Feat Red Misc	Feat Red Acc	NN Misc	NN Acc
0	Other	10808	327	96.974463	304	97.187269	510	95.281273
1	back to news	2458	199	91.903987	224	90.886900	261	89.381611
2	facebook in background for 200 sec	1470	200	86.394558	169	88.503401	268	81.768707
3	first conversation selection	1941	23	98.815044	11	99.433282	44	97.733127
4	friends selection	4640	45	99.030172	36	99.224138	49	98.943966
5	menu message selection	984	41	95.833333	32	96.747967	82	91.666667
6	menu news selection	1024	12	98.828125	7	99.316406	69	93.261719
7	menu selection	3796	259	93.177028	295	92.228662	503	86.749210
8	notification friends selection	1716	75	95.629371	50	97.086247	153	91.083916
9	notification general selection	1030	101	90.194175	82	92.038835	111	89.223301
10	open facebook	3800	124	96.736842	109	97.131579	239	93.710526
11	post button selection	1574	83	94.726811	84	94.663278	82	94.790343
12	send message selection	1608	40	97.512438	27	98.320896	137	91.480100
13	status post selection	1301	48	96.310530	33	97.463490	143	89.008455
14	user alfa selection	1913	26	98.640878	31	98.379509	66	96.549922
15	user profile selection	1567	94	94.001276	79	94.958519	137	91.257179
16	write selection	1269	17	98.660362	9	99.290780	46	96.375099
17	writing message	3776	63	98.331568	45	98.808263	141	96.265890
18	writing search	2380	25	98.949580	33	98.613445	74	96.890756
19	writing to alfa	1264	57	95.490506	30	97.626582	97	92.325949

Facebook

	Action	Number of sample	Misclassification	Accuracy	Feat Red Misc	Feat Red Acc	NN Misc	NN Acc
0	Other	596	103	82.718121	117	80.369128	184	69.127517
1	Twitter in backgroud	2951	274	90.715012	236	92.002711	273	90.748899
2	back to home	507	101	80.078895	87	82.840237	154	69.625247
3	back to me	2559	69	97.303634	46	98.202423	233	90.894881
4	contact selection	2138	20	99.064546	27	98.737138	80	96.258185
5	direct messages selection	2873	15	99.477898	10	99.651932	56	98.050818
6	discover selection	2689	114	95.760506	96	96.429900	310	88.471551
7	edit selection	917	33	96.401309	20	97.818975	52	94.329335
8	first message selection	1154	39	96.620451	15	98.700173	42	96.360485
9	followers selection	1009	50	95.044599	32	96.828543	313	68.979187
10	following selection	929	19	97.954790	26	97.201292	61	93.433800
11	me selection	2134	60	97.188379	55	97.422680	153	92.830366
12	new tweet selection	1021	20	98.041136	15	98.530852	299	70.714985
13	open Twitter	6278	91	98.550494	102	98.375279	207	96.702772
14	refresh home	1992	9	99.548193	5	99.748996	39	98.042169
15	send selection	1372	7	99.489796	10	99.271137	50	96.355685
16	send tweet selection	1467	75	94.887526	100	93.183367	124	91.547376
17	tweets selection	2221	27	98.784331	13	99.414678	81	96.352994
18	writing tweet	1452	130	91.046832	108	92.561983	242	83.333333

Twitter

	Action	Number of sample	Misclassification	Accuracy	Feat Red Misc	Feat Red Acc	NN Misc	NN Acc
0	Other	7674	112	98.540526	50	99.348449	159	97.928069
1	background for 200 seconds	2032	247	87.844488	214	89.468504	404	80.118110
2	done note edit	2268	58	97.442681	30	98.677249	210	90.740741
3	market selection	2151	19	99.116690	6	99.721060	68	96.838680
4	new note title input	1286	52	95.956454	46	96.423017	92	92.846034
5	open evernote	1603	4	99.750468	15	99.064255	58	96.381784

Evernote

References

[1] : *M. Conti, L. Mancini, R. Spolaor, N. Verde (2014). Can't you hear me knocking: Identification of user actions on Android apps via traffic analysis;*
[2] : *Dataset https://spritz.math.unipd.it/projects/analyzinguseractionsandroid/*