Dear Hiring Manager and team members,

Hi! My name is Chin-Ting Ko, I'm actively looking for data engineer position.
I recently graduated from Johns Hopkins University majoring Computer science in data science and cloud computing track, currently live in bay area. Below is the solutions and analysis report for the coding assignment.

Feel free to let me know if any questions and suggestions, thank you!

Chin-Ting Ko

Deliverable

. 1) filename of training data

data_coding_exercise.txt

. 2) filename of test data

test_data_coding_exercise.txt

. 3) filename of prediction results from test data

prediction_results.txt

Executable script

python uaML.py

Background

User Agent String: When I first read through the assignment, the user-agent strings is something new to me, so I spend some time go through concept of the user agent. The main idea is to notify server hosting your browser and system details. I can imagine it's really practical assignment which likely in daily tasks.

Some of the key information from user agent strings such as Browser, Browser type, Version,Device, OS etc.

Reference:
https://msdn.microsoft.com/en-us/library/ms537503(v=vs.85).aspx
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent

Approach and Data Analysis

When I started the assignment, I feel it's another data parsing/formatting task. Yes and No, unfortunately user string is not well formatted, so it make sense use machine learning technique classification to help predict result, in this assignment is to predict browser type and version.

As always, I started to analyze the data set as beginning. I listed below result of term frequency, and it's with several different browser types, but not evenly distributed. Most user using Chrome or Chrome Mobile, which might bias the training model. In the code, for each browser family I simply use training data up to 5000 as maximum.

User Family Data Brief: (Total 421215 data sets)

AOL 100
Chrome 308656
Maxthon 107
Firefox Mobile 248
QQ Browser 365
Android 8155
QQ Browser Mobile 1745
AppleMail 120
BlackBerry WebKit 275
Mobile Safari 2383
Sogou Explorer 95
Facebook 18803
Amazon Silk 224
Safari 626
IE Mobile 305
Chrome Mobile iOS 911
Opera Mini 626
UC Browser 5595
Opera 921
YandexSearch 111
Chrome Mobile 50095
Edge Mobile 116
Opera Mobile 1281
Puffin 75
Firefox iOS 73
IE 17426
Firefox 1661
Edge 117

Before I feed any data to training model, it would be nice to parse data which easier for later data manipulation and analysis. I tried to categorize the data to header data (Mozilla), System information, Platform, Platform details, and Extensions. My initial thought the browser information is mainly from Extensions, and Systems. Others data are irrelevant so it's better remove to increase the training model accuracy, and possibly improve computing speed.

The coding assignment I've finished is python, sklearn as machine learning tool. (Naive Bayes, and SVM)

First try use user agent extensions as training features. For high frequency browser type, I set 5K as upper limit to prevent training model predict same result. However later on experiments shows this extra trick is not necessary, so I just remove it from code.

User Family

DataSet	Features	Classifier	Precision	Recall	F1 Score
all 421215	all UA strings	MultinomialNB	0.92	0.92	0.91
all 421215	Extensions+ System	MultinomialNB	0.92	0.92	0.91
all 421215	Extensions	MultinomialNB	0.93	0.94	0.93
first 10000	all UA strings	MultinomialNB	0.86	0.89	0.85
first 10000	Extensions+ System	MultinomialNB	0.86	0.89	0.86
first 10000	**Extensions**	**MultinomialNB**	**0.87**	**0.90**	**0.87**
all 421215	all UA strings	BernoulliNB	0.91	0.93	0.92
all 421215	Extensions+ System	BernoulliNB	0.91	0.91	0.90
all 421215	**Extensions**	**BernoulliNB**	**0.95**	**0.94**	**0.94**
first 10000	all UA strings	BernoulliNB	0.82	0.86	0.82
first 10000	Extensions+ System	BernoulliNB	0.82	0.83	0.80
first 10000	Extensions	BernoulliNB	0.83	0.87	0.84
first 10000	all UA strings	SVM	0	0.06	0.01
first 10000	Extensions+ System	SVM	0	0.06	0.01
first 10000	Extensions	SVM	0	0.06	0.01

User Version

DataSet	Features	Classifier	Precision	Recall	F1 Score
all 421215	all UA strings	MultinomialNB	0.96	0.96	0.95
all 421215	Extensions+ System	MultinomialNB	0.96	0.96	0.96
all 421215	Extensions	MultinomialNB	0.93	0.93	0.93
first 10000	all UA strings	MultinomialNB	0.92	0.94	0.93
first 10000	Extensions+ System	MultinomialNB	0.92	0.94	0.93
first 10000	Extensions	MultinomialNB	0.90	0.92	0.90
all 421215	all UA strings	BernoulliNB	0.96	0.96	0.96
all 421215	Extensions+ System	BernoulliNB	0.96	0.96	0.96
all 421215	**Extensions**	**BernoulliNB**	**0.96**	**0.95**	**0.95**
first 10000	all UA strings	BernoulliNB	0.92	0.93	0.92
first 10000	Extensions+ System	BernoulliNB	0.93	0.95	0.93
first 10000	**Extensions**	**BernoulliNB**	**0.92**	**0.93**	**0.93**
first 10000	all UA strings	SVM	0.55	0.74	0.63
first 10000	Extensions+ System	SVM	0.55	0.74	0.63
first 10000	Extensions	SVM	0.55	0.74	0.63

Observation

Obviously SVM is not a good machine learning tool for text classification, Naive Bayes performs way better. The reasons is SVM is better for numbers predictions, and Naive babes is better for text classification.

In general BernoulliNB performs slightly better than MultinomialNB, the reason is likely due to 0/1 (exist/non-exist) nature of user agent strings. To consider efficiency, I chose BernoulliNB only with “Extensions” features for training model. It performs pretty well even with less data sets, precision is around 90~95% for both user family and user version.

Further thought & Future Work

Once I wrap up this coding assignment, I feel machine learning tool maybe is not necessary since the browser information is actually already included in the strings. Maybe a parser can pretty much do the work. I started to research many user agent parser. I realized it’s already a matured parser package out there, but I feel the effort is way heavier than I thought in very beginning. The advantage of the machine learning tool for user agent is less code with good enough precision, but the parser probably is almost 100% prevision rate which is the reason why parser seems more popular.

Appendix

Sample output:

	precision	recall	f1-score	support
AOL	0.00	0.00	0.00	100
Amazon Silk	1.00	0.04	0.08	224
Android	0.91	0.90	0.91	8155
AppleMail	0.00	0.00	0.00	120
BlackBerry WebKit	0.00	0.00	0.00	275
Chrome	1.00	0.98	0.99	308656
Chrome Mobile	0.86	0.94	0.90	50095
Chrome Mobile iOS	1.00	0.98	0.99	911
Edge	0.00	0.00	0.00	117
Edge Mobile	0.00	0.00	0.00	116
Facebook	1.00	0.72	0.84	18803
Firefox	0.86	0.96	0.91	1661
Firefox Mobile	0.00	0.00	0.00	248
Firefox iOS	0.00	0.00	0.00	73
IE	0.64	1.00	0.78	17426
IE Mobile	0.33	0.00	0.01	305
Maxthon	0.00	0.00	0.00	107
Mobile Safari	0.91	0.96	0.94	2383
Opera	0.45	0.41	0.43	921
Opera Mini	0.82	0.84	0.83	626
Opera Mobile	0.87	0.95	0.91	1281
Puffin	0.00	0.00	0.00	75
QQ Browser	0.99	0.84	0.91	365
QQ Browser Mobile	0.97	0.97	0.97	1745
Safari	0.56	0.28	0.37	626
Sogou Explorer	0.00	0.00	0.00	95
UC Browser	0.98	0.98	0.98	5595
YandexSearch	0.00	0.00	0.00	111
avg / total	0.96	0.95	0.95	421215

Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.36 (KHTML like Gecko) Chrome/39.0.2195.31 Safari/537.36 Chrome 39 Chrome 39
Mozilla/5.0 (Linux; Android 4.1.1; GT-N8010 Build/JRO03C) AppleWebKit/537.36 (KHTML like Gecko) Chrome/54.0.2840.85 Safari/537.36 Chrome 54 Chrome Mobile 54
Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML like Gecko) Chrome/39.0.2175.61 Safari/537.36 Chrome 39 Chrome 39
Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML like Gecko) Chrome/45.0.2480.83 Safari/537.36 Chrome 45 Chrome 45
Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.36 (KHTML like Gecko) Chrome/37.0.2072.40 Safari/537.36 Chrome 37 Chrome 37
Mozilla/5.0 (Windows NT 5.1; Win64; x64) AppleWebKit/537.36 (KHTML like Gecko) Chrome/45.0.2469.28 Safari/537.36 Chrome 45 Chrome 45
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML like Gecko) Chrome/33.0.1808.66 Safari/537.36 Chrome 33 Chrome 33
Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML like Gecko) Chrome/41.0.2296.3 Safari/537.36 Chrome 41 Chrome 41
Mozilla/5.0 (Windows NT 5.1; Win64; x64) AppleWebKit/537.36 (KHTML like Gecko) Chrome/35.0.1975.31 Safari/537.36 Chrome 35 Chrome 35
Mozilla/5.0 (Windows NT 6.2; Win64; x64) AppleWebKit/537.36 (KHTML like Gecko) Chrome/42.0.2349.65 Safari/537.36 Chrome 42 Chrome 42