Purdue University

Dataset Generation and Detection for False Information

Fault-Tolerant Computer System Design
ECE 695 / CS 590

Lee, Wen-chuan (lee1938)
Ling, Tzu Hsuan (ling21)
Tsai, Shin-yeh (tsai46)

December 20, 2015

Table of Contents

List of Tables and Figures

1. Problem Motivation

Data has become increasingly important throughout the past decade. As we move towards the digital era, more people realize the significance of processing and analyzing data. As a type of asset, the accuracy of what the collected data entails determines the trustworthiness of the individual or organization responsible for the data. There have been several types of data such as weather and crime rate; to limit our scope for this project, we have chosen to focus on identifying potentially dishonest survey data submissions.

Surveys have been digitized in the recent days. In the past, most surveys conducted were paper-based. One of the advantages of paper-based survey is that most sessions are supervised, and participants are more willing to answer truthfully to the questions under such condition. Paper-based surveys, however, have one huge drawback - it is difficult to distribute the surveys to wide range of audiences. As the cloud era emerges, people have begun to convert surveys into digital format, where they easily share the forms through several channels such as social media. This way, survey forms can reach more people, yet the accuracy can be significantly lowered because of the drawback inherited from unsupervised survey sessions.

Our goal of this project is to experiment methods of identifying dishonest data submissions to digital surveys, and then further develop on our implementations and research into ways to improve the accuracy of alienating falsey user input.

2. Problem Formulation

To gather the survey data, the survey initiator need to ask several users to fill out the questionnaire. However, there lies a problem where it can be hard to detect false submissions because most surveys are anonymous and users can decide whether to fill out the true or false information freely.

To help detect such false information, people have come up with trap questions in the questionnaire. Although these types of questions can help, the effect is limited. Furthermore, users tend to dislike like long questionnaires, and having multiple questions attributed to the same meaning contributes to the overall length of the survey. This may demotivate participants from fully completing the survey.

To solve the problem of false or dishonest survey submissions, we choose to use electronic based survey implementing well-designed questions. The reason we use electronic based survey is that we can track and record users' behaviours while they answer the questions, under our assumption that users' behaviours and the reliability of questionnaires are highly correlated.

To observer users' behaviours, we use a well-designed website to record users' behaviours while answering questions such as the amount of time spent on each question and mouse movement. After recording the user's behaviour for each question, we utilize machine learning to determine the correlation between users' behaviours and the reliability of submissions, through well-known techniques such as naive Bayes classifier, decision tree, and SVM. This helps us classify and isolate false submissions from the correct ones.

3. Solution Approach

Few decades ago, paper-based questionnaires dominated the world. Almost all of the surveys are conducted using pen and paper. Since paper based questionnaires only record answers to each question, researchers could only validate the reliability of a questionnaire by designing trap questions and apply democracy to the submitted answers. Recently, digital surveys have become popular due to cloud technologies. In comparison with traditional paper-based surveys, digital surveys is capable of providing additional features for validating the inputs. For example, while paper-based surveys cannot precisely track where the answers are submitted, digital surveys enable the ability to validate data submissions through IP addresses. Therefore, researchers have started thinking about which new features can be added to help validate data submissions.

In this project, we focus on behaviour analysis. In traditional surveys, behaviours of users could not be easily tracked. This will be a new feature in digital surveys, and our hypothesis is that user behaviour is highly related to the reliability of an answered questionnaire. We have built a website with a sophisticate questionnaire; while users are answering the survey, we not only record their answers, but also track the mouse movement and time they spent on answering in each question. The answering order is also recorded in our web application.

In order to determine a set of well-designed questions that fit our approach, we have studied several papers in the psychology field of study and selected a publication that lists questions related to job and work involvement[1]. In order to fully focus on users' behaviours, we chose a relatively short questionnaire to incentivize users to complete the questionnaire.

The questions, all in the form of Likert scale, are listed in as follows.

1. "The most important things that happen to me involve my present job"
2. "To me, my job is only a small part of who I am"
3. "I am very much involved personally in my job"
4. "I live, eat and breathe my job"
5. "Most of my interests are centered around my job"
6. "I have very strong ties with my present job which would be very difficult to break"

[1] Measurement of job and work involvement. By Kanungo, Rabindra N. Journal of Applied Psychology, Vol 67(3), Jun 1982, 341-349

7. "Usually I feel detached from my job"
8. "Most of my personal life goals are job-oriented"
9. "I consider my job to be very central to my existence"
10. "I like to be absorbed in my job most of the time"

We group the questions into two reversed question groups. Questions 2 and 7 are in one group and the other questions are in the other group. Reversed question groups are groups of questions asking the same questions with completely opposite phrasing; therefore, the answers to each group are expected be on the opposite ends of a Likert scale. If there exists a contradiction between these two groups, for example, user answering “Strongly Agree” for question 1 and “Strongly Agree” again for question 2, the likelihood that the user is dishonest will be high. Such submission is thus likely to be labelled as dishonest.

In addition to the aforementioned questions, there are three extra short-answer questions:

1. The user’s date of birth (at the beginning of the questionnaire)
2. Location where the user resides
3. The user’s date of birth (at the end of the questionnaire)

The second question is not related to the reliability, yet if a user fails to remain consistent between the two date of birth questions, we classify this questionnaire as false submission. Since the answering order is also tracked in our solution, if irregular order is observed from a submission, it is also labelled as dishonest.

Our goal is to validate our hypothesis whether user behaviours are related to the reliability of a questionnaire or not. Therefore, after collecting all the features, we label the data by the above reliability analysis including trap questions, reversed questions and answering order. We then train our learning model by these labelled data accordingly. We have adopted several training models in our approach, which will be explained in section 4.2.

4. Implementations

Our implementation involves two core elements. One of which is a digital survey that incorporates questions and metrics from the aforementioned psychology research paper. The questions contain traps and mechanisms to help identify dishonest user submissions. The other core element of our implementation is a machine learning program that mines through the collected survey responses and categorize them into honest or dishonest data.

The source code of our digital survey and the machine learning process has been uploaded to our Github[2].

4.1 Digital Survey

The digital survey is a web application hosted on a cloud service. We implemented the application with some open source libraries which are simply helpers to help serialize and storing data.

4.1.1 Technology Stack

The web application is deployed on and hosted by Heroku[3] hosting service. The main reason we chose Heroku as our hosting provider is that the service allows simple and straightforward deployment of our application. For instance, we can simply connect our Heroku account to our Github repository and enable automatic pulling. When this feature is enabled, each time we merge new code into our master branch, Heroku will initiate a pull operation from the remote and rebuild the application stack, including managing application dependencies.

The server-side language we selected is Node.js. This particular language is used for the wide availability of open source libraries across npm[4], the dominant package manager for Node.js. The libraries we used for server-side programming include:

1. Koa.js (http://koajs.com) - a framework for Node.js that provides intuitive exception handling and middleware management
2. koa-router (https://github.com/alexmingoia/koa-router) - the router middleware for directing HTTP requests to appropriate handlers

[2] Source code on Github - https://github.com/thling/fts-survey
[3] Heroku hosting service - https://heroku.com/
[4] npm package manager - https://www.npmjs.com/

3. koa-generic-session (https://github.com/koajs/generic-session) - session management middleware
4. Extended JavaScript (EJS, http://ejs.co) - front-end user interface rendering that use injected JavaScript in HTML source code to render otherwise complicated to implement HTML DOM elements
5. lodash (http://lodash.com) - a JavaScript library consisted of a large set of frequently used functions
6. co (https://github.com/tj/co) - ECMAScript 6 generator function[5] wrapper library for JavaScript applications
7. mongoose (http://mongoosejs.com/) - MongoDB object data mapper for enhancing non-strict database schema modelling

Because Heroku purges memory after 30 minutes of inactivity with the free-tier service, we opted to offload data handling and storage to third-party database hosting - MongoLab. As the name suggests, the underlying database is MongoDB, which we decided to adopt because of the NoSQL, document-store characteristics. The primary benefit of using such type of database is that the stored data are already denormalized when exported, as opposed to traditional relational databases that enforce strict schemas and data types in addition to the heavy focuses on normalizing data into tables and columns. With naturally denormalized data and a database system designed to efficiently process them, the ability to export as JSON format makes our data mining and machine learning process intuitive.

Lastly, the main element of the digital survey is the front-end system. The client-side elements are built with HTML, CSS, and JavaScript. As mentioned above, HTML is rendered by Extended JavaScript where we can inject JavaScript code into our HTML files to reduce complexity and help modularising the program (for instance, we can use loops to generate `<li>` elements for all items in an array passed to the EJS rendering engine). For CSS, a helper library named Materialize[6] is utilized to help enhancing the aesthetics of the website. Lastly, for JavaScript we adopted the jQuery[7] library that provides useful functions that help simplify complex code such as AJAX calls and DOM element selectors, while providing

[5] ECMA 6 Generator Functions - http://mzl.la/1gSa8Gu
[6] Materialize - http://materializecss.com/
[7] jQuery - http://jquery.com/

excellent interfaces for Promise-based design pattern, which allow us to avoid the infamous Callback-Hell[8] usually observed in callback based function calls.

4.1.2 Questions Recording and Mouse Tracking

To record the users' behaviours at greater granularities, we decided to implement folding questions (Figure 1) so that each user can only view contents of one question at a time.

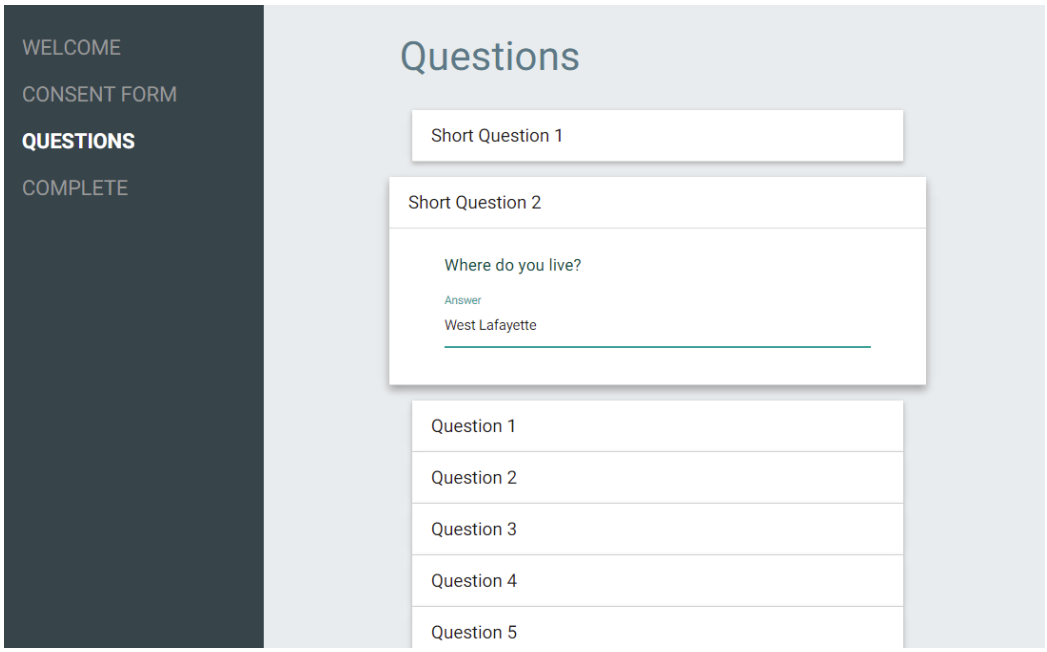

Figure 1. Folding questions

The questions list include a submit button at the end of the page, yet each time users switch to a different question, all recorded behaviours are sent to the server to preserve the answering order and give users as little chance as possible to modify the tracked data. This is achieved by associating a pair of server-side generated time value, formatted in epoch time, with each of the uploaded answers (later referred to as a "tracker"). Table 1 explains the fields in each of the trackers.

Field Name	Descriptions
Question ID	The ID of the question this tracker is tracking
Start Time	The time in epoch when users begin to look at a question. This is set as soon as users switched to the currently viewing question

[8] Callback-Hell - http://callbackhell.com/

End Time	The time in epoch when users stop looking at a question. This is set as soon as users switched to another question
Answer	An object with two fields: “new” and “old.” “new” indicates the new answer the user provides, and “old” indicates the last answer the user entered. If this is the first time the user answered a particular question, the “old” field will be empty
Mouse	An array of 2-tuples. Each tuple represent a pair of window coordinates relative to the top left of the browser window. The tuples are the pixels where the mouse have ever moved to

Table 1. Explanation of the use of each field in tracker element

Mouse tracking is implemented using JavaScript’s mouse event listener. We created a transparent `<canvas>` element that covers the entire browser window and register the listener and corresponding handler to the canvas. Using a dedicated canvas element for the purpose not only helps isolate out special features to a specific DOM element but also allows easier debugging when we need to make sure mouse movements are recorded correctly - we can easily connect two pairs of coordinates together with a line on the screen by colouring the pixel on the canvas. Figure 2 illustrates how canvas helps in recording mouse data as well as debugging.

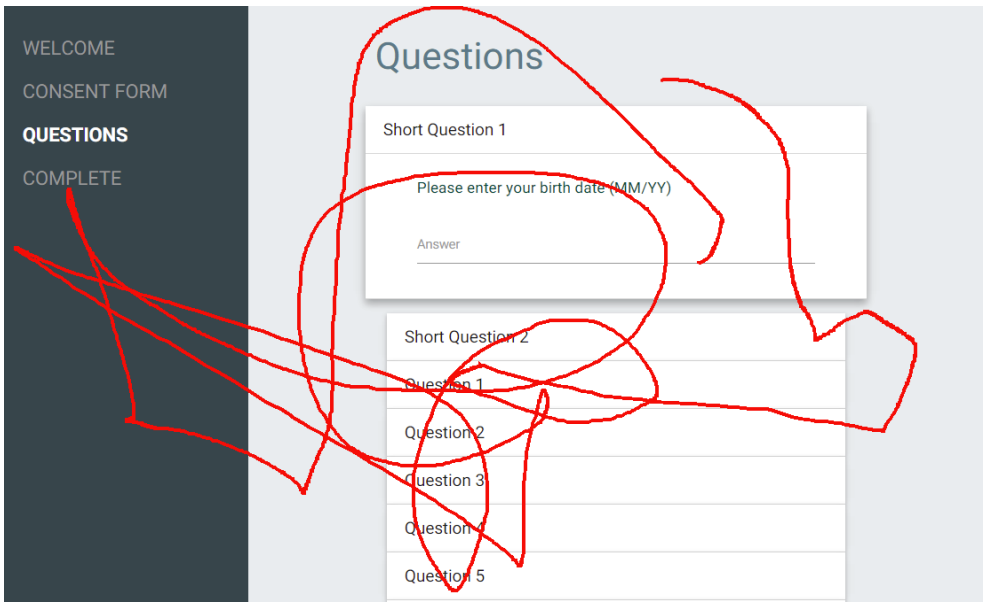

Figure 2. Illustration of mouse movements (red line is mouse trace)

Each time the mouse is moved, the coordinates associated with the event object is stored in the browser memory. As previously discussed, the records are submitted to the server as soon as the users switch to different questions.

4.2 Machine Learning

This section describes the dataset and machine learning models used in this project. This dataset contains 42 records which are collected via the above web application.

4.2.1 Data, Features, and Labelling

Each data contains the following features:

1. Answering time for each question
2. Total moving distance based on mouse traces in each question
3. The displacement between the initial point and the end point in each question

Labelling of data is completed by answering order and reliability analysis from reversed questions and trap questions which is mentioned in section 3. If an answer fails to pass the validation of reversed questions and trap questions, it is labelled as a false submission. Otherwise, it is labelled as a true submission.

4.2.2 Machine Learning Algorithms

Although we collect answering orders from the web application, we do not feed the data into our learning model. This is because our goal is to look for relationships between users' behaviours and reliability of survey submissions. If answering order were fed into the training process, it will cause the model to learn how we use the data to label the survey response, and the classification will become highly correlated to the answering order. On the other hand, if the goal of our study were to further increase the accuracy of the classifying mechanism, answering order can then be used as a feature for the training model.

The machine learning procedures are implemented in Python 2.7.1. The program utilizes the following libraries:

1. numpy, 1.8.2
2. sklearn, 0.17

3. pandas, 0.17.1

We test with seven machine learning algorithms:

1. Nearest Neighbor
2. SVM with linear kernel
3. SVM with RBF Kernel
4. Decision Tree
5. Random Forest
6. Adaboost
7. Naive Bayes

and with three different folding algorithms:

1. Kfold
2. Stratified Kfold
3. Leave One Out

5. Evaluation

The results of our experiments are included in the table listed in Appendix A. This section explains the observations we made from the table.

It is conspicuous to see that naive Bayes classifier (NBC) outperforms the techniques. This is because the assumption that NBC makes fits into our hypothesis extremely well. NBC assumes that all features are conditionally independent to each other. If a user decides to submit false information, the collected features of each question are independent since the user does not read the descriptions of the question, and thus answering one question does not affect the way the user answer other questions. As a result, given the submission is false, all the features are conditionally independent to each other, and this fits NBC's assumption well.

Because there are 39 features in our training model, it is difficult to show the $\boldsymbol{\theta}$ and $\boldsymbol{\sigma}$ of all the features. We have attached all the details of the results from the NBC learning model as part of our source code submissions, under the MachineLearning archive.

The performance of Support Vector Machine (SVM) is not as good as NBC. It is barely better than random guessing. We believe that this is because we did not normalise mouse movement while calculating the distances and displacements. The distances between different browser window sizes will cause the coordinates of the collected vectors to be different, and the difference significantly impedes the performance of SVM. This issue will be solved as a future work to further improve the performance of our learning model.

Although the Leave One Out folding algorithm performs slightly better than the others, the differences in performance are not significant among all folding algorithms. The reason is that our sample size is too small. The Leave One Out algorithm was given the largest portion of training data. Once we collect more samples, the differences among these folding algorithms may become more significant since both Kfold and stratified Kfold can receive enough samples to train the respective learning models.

6. Future Work

There are a few areas where further studies and developments can focus on. We discuss a few of those in this section.

6.1 New Features

Currently we only record users' answering time, discrepancies in answers, and mouse movements for each question. As a result, we only have three features to use for the learning phase. Although the result of NBC proves to be better than the others, some of the algorithms only perform as good as random guessing. To help improve the performance and available features, we can consider recording more types of behaviours and develop new features from the data.

6.2 Sensitive Questions

Our questionnaire does not contain any sensitive questions such as gender, and race. However, using these types of questions may help us better detecting false submissions as some groups of users with similar attributes are more likely to tell lies on specifics topic. Therefore, asking sensitive questions may help us detect false information more accurately.

6.3 Data Normalisation

Our current data collection is performed through recording user behaviours on our custom-designed website. However, users may use different devices or browsers to complete the survey, and this can create discrepancies on the collected data, which can be detrimental to the performance of the classification algorithms. In the future, information on the screen size, device type, and other unique attributes can be recorded to help normalizing vector data.

7. Conclusions

Digital surveys can be used to detect false submissions by combining user behaviour recording and machine learning. According to our experiment, the accuracy of NBC is approximately 85%, and both type 1 and type 2 errors are extremely low at approximately 7% and 4% respectively. We believe that in the big-data era, user behaviour analysis is indispensable for false information detection.

Appendix A - Quantitative Results of Experiments

classifier	fold	accuracy	precision	recall	fscore	type1Error	type2Error
Nearest Neighbors	stratifiedKFold	0.5476190476	0.4782608696	0.6111111111	0.5365853659	0.2857142857	0.1666666667
Nearest Neighbors	KFold	0.5714285714	0.5	0.6111111111	0.55	0.2619047619	0.1666666667
Nearest Neighbors	LeaveOneOut	0.5	0.4285714286	0.5	0.4615384615	0.2857142857	0.2142857143
Linear SVM	stratifiedKFold	0.5952380952	0.5238095238	0.6111111111	0.5641025641	0.2380952381	0.1666666667
Linear SVM	KFold	0.5714285714	0.5	0.6666666667	0.5714285714	0.2857142857	0.1428571429
Linear SVM	LeaveOneOut	0.5952380952	0.5217391304	0.6666666667	0.5853658537	0.2619047619	0.1428571429
RBF SVM	stratifiedKFold	0.5714285714	0	0	0	0	0.4285714286
RBF SVM	KFold	0.4285714286	0.125	0.0555555556	0.0769230769	0.1666666667	0.4047619048
RBF SVM	LeaveOneOut	0.5714285714	0	0	0	0	0.4285714286
Decision Tree	stratifiedKFold	0.5952380952	0.5294117647	0.5	0.5142857143	0.1904761905	0.2142857143
Decision Tree	KFold	0.6428571429	0.5789473684	0.6111111111	0.5945945946	0.1904761905	0.1666666667
Decision Tree	LeaveOneOut	0.5476190476	0.4736842105	0.5	0.4864864865	0.2380952381	0.2142857143
Random Forest	stratifiedKFold	0.4761904762	0.3888888889	0.3888888889	0.3888888889	0.2619047619	0.2619047619
Random Forest	KFold	0.4761904762	0.3888888889	0.3888888889	0.3888888889	0.2619047619	0.2619047619
Random Forest	LeaveOneOut	0.6428571429	0.5882352941	0.5555555556	0.5714285714	0.1666666667	0.1904761905
AdaBoost	stratifiedKFold	0.5952380952	0.5294117647	0.5	0.5142857143	0.1904761905	0.2142857143
AdaBoost	KFold	0.7380952381	0.7058823529	0.6666666667	0.6857142857	0.119047619	0.1428571429
AdaBoost	LeaveOneOut	0.6904761905	0.619047619	0.7222222222	0.6666666667	0.1904761905	0.119047619
Naive Bayes	stratifiedKFold	0.8333333333	0.7894736842	0.8333333333	0.8108108108	0.0952380952	0.0714285714
Naive Bayes	KFold	0.8333333333	0.8235294118	0.7777777778	0.8	0.0714285714	0.0952380952
Naive Bayes	LeaveOneOut	0.880952381	0.8421052632	0.8888888889	0.8648648649	0.0714285714	0.0476190476