AI Lie Detector

COMPGC27 – Programing for Business Analytics

Group Project Report

Group 2

16129877 – Leila Illanes Mayta – uczllil@ucl.ac.uk
17028872 – Weisi Han – uczlwha@ucl.ac.uk
17137702 – Xing Heng – ucabxh3@ucl.ac.uk
17114792 – Jenny Yang – jenny.yang.17@ucl.ac.uk
17102418 – Jorge Alarcon – jorge.diaz.17@ucl.ac.uk
17105731 – Diego Camacho – diego.carbajal.17@ucl.ac.uk

1 Introduction

Lie detection is not a new idea but rather a technique that humans have always wished to obtain, but yet to discover the best one. In the past decades, there has been plenty of research of different methods, leading to many products for the market. To list a few, people have tried to use polygraph, voice stress analysis, brain observations electroencephalography (EEG) and even drugs in the past. However, none of these have been very reliable in telling lies and truths apart. A well-trained person could easily deceive the method, especially when lying about something less important (i.e. white lies). Today, lie detection is mainly being used in law enforcement, and the "most reliable" method being used has mostly remained unrevealed to avoid educating the criminals.

Micro-facial expression recognition has evolved over the last couple of years and has been reliable enough to build intelligent systems on top of them. Many researches have suggested that frequency, duration and intensity in micro-facial expressions, using the Facial Action Coding System (FACS), can function as a good signal of deception. Some gestures are simply hard to control when one is nervous lying or tend to be overdone when the deceivers are trying too hard to hide.

Given this, we decided to create and develop an interactive web application that uses this theory to detect lies, using micro-facial expression recognition technology and machine learning algorithms. Our goal is to allow people to have access to lie detection technology using only their laptops, without having to invest in any special equipment. We believe that there is a huge business potential for this application, as this technology can be utilized in many different areas such as education, recruitments, consulting, etc.

2 Methodology

AI Lie Detector is an interactive web application that uses machine learning algorithms to detect whether a person is telling the truth or a lie. For this, we use the commonly played ice breaker game Two Truths and A Lie. The application will present 3 questions to a person, who has to answer two with the truth and one with a lie. Then, our AI Lie Detector will try to guess in which question the person lied on using micro-facial expression recognition technology.

Using this game-like setup, we are capturing and storing a lot of expression data from people playing it, allowing us to train a machine learning algorithm which will lead to it being able to pick up the main characteristics of a lying person, thus improving in its ability to detect a liar.

Approaching this product development in an evolutive way, this stage is going to be regarded as phase zero where we created the baseline of the application, and further improvements will come in the form of context-aware questionnaire videos, data acquisition analysis and improved model selection after more data has been accumulated.

The design and development of the phase zero of AI Lie Detector has been done using python, and can be separated in three main parts: the front end, the data storage and the modelling.

2.1 Front end

The front end of AI Lie Detector consists in a simple web page that allows interaction between the application and the user. It uses Flask, a simple python web framework, to display and handle all the data and events.

It starts with an initial page that displays the instructions of the game, a button that initializes the camera to do the real-time capturing of facial expressions, and a button to start the game. When the page is loaded, we assign a unique session id and retrieve 3 random questions from the database. More detail about this part will be explained in the Data Storage section.

For the facial expression analysis, we use two existing public and free APIs: Facial Expression Analyzer by Affectiva[1] (affdex) and Eye Gazer Tracker by xLabs[2] (eyetracker). Both of them uses the camera (no mic) to do real-time analysis, and return the results in a table format with timestamps. The affdex API gives you, for each timestamp, a list of facial features and expressions, such as joy, anger, smile or openness of mouth (full list in the appendix), with a number between 0 and 100 that indicates how present is the feature in the person's face. Similarly, the eyetracker API give you, for each timestamp, the coordinates within the screen where the person was looking.

In terms of privacy, it is important to note that we send only image data (no audio) to these API, and that we nor them store any video or picture but only text data with the results.

[1] https://affectiva.readme.io/docs/using-the-emotion-sdk

[2] http://xlabsgaze.github.io/docs/home.html

Once the user has started the camera and the connection is made between the AI Lie Detector and both API, the "Give it a try!" button activates.

Once started, the user will be presented with 3 questions. For each question, they will have 3 seconds to read the question and then 10 seconds to make an answer. This is important, because although we store the data from the whole 13 seconds per question, only data collected within the 10 second window to answer will be used in our analysis. We have figured that the facial expressions between questions are not accurate, as there are possibilities that the user may have not "reset" their emotions from the last question or that they are laughing at the new question as they simply find it funny.

Once finished, we retrieve all the information from the affdex and eyetracker API and store it in the database using the session id. Also, we send it to a predictive function that uses a pre-trained model to guess in which question (1, 2 or 3) the user lied on. We then store this prediction in the database.

Lastly, the user is presented with the 3 questions he was asked, and the one the model predicted as a lie colored differently. Also, the app will ask the user to confirm in which question he actually lied on. This answer allows us to correctly "label" our data with the correct answer, thus helping in the collection of data for training purposes.

The Figure 1 is a simplified diagram showing the flow of our application. The detailed sequence diagram can be found in the Appendix A.

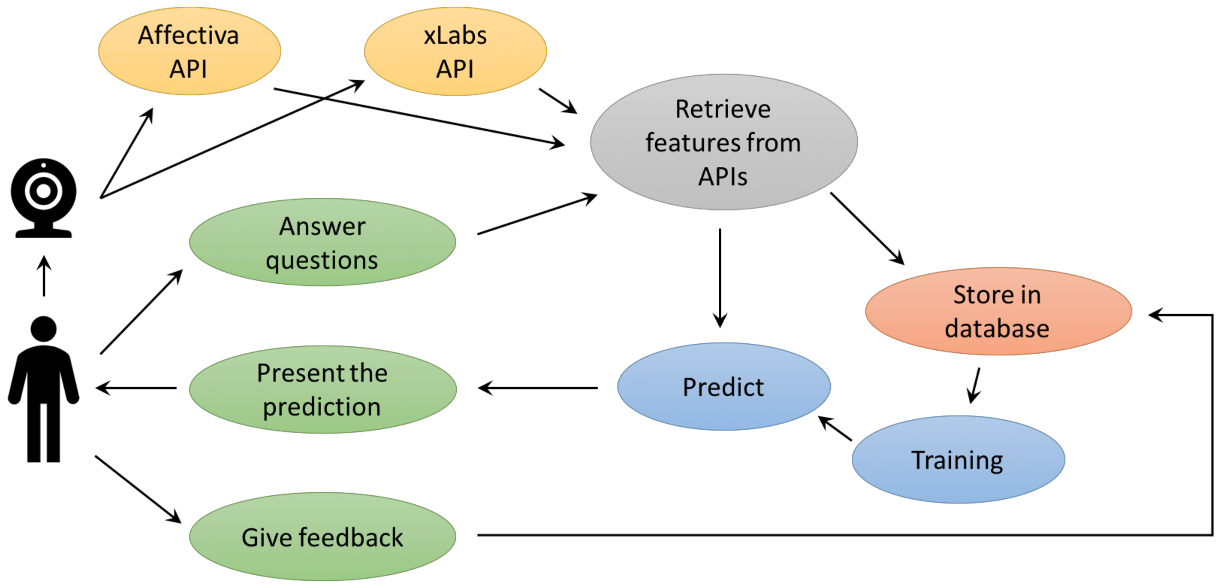

Figure 1: Use case diagram of the AI Lie Detector.

2.2 Data storage

For the data storage, we use a mysql database and a python script to process it and interact with the database engine.

The database is composed of six tables: Question, Session, Session_question, LogEvent, Expression, and Gaze. Except for Question table, all the other tables are real-time updated, which means once a piece of new data is generated and collected it will be inserted into a specific table. Among the tables, LogEvent, Expression and Gaze are used for storing user activities/features data captured by the APIs. The Entity Relationship diagram is shown in Figure 3 which illustrates the relationship between entities and the attributes of each entity.

Question: There are 23 test questions stored in this "Question" table (see Figure 4) with each assigned a question ID. This table is static and will not be frequently changed. The three questions in each test will be randomly selected from this table.

Session: Once a user enters the system and conducts a test, we call it a session. This "Session" table (see Figure 5) stores the ID of all user Sessions along with its *timestamp* (the time the user starts doing the test).

Session_question: There are 3 questions within each session, and each question is a unit to train the model. This "Session_question" table (see Figure 6) keeps an account of the three questions in each session (by assigning each question a unique *sessionQuesID*) together with its corresponding *sessionID*, *questionID*, *sequence* (the order of the three questions in a test, with a value equaling to 1,2 or 3), *pLabel* (the answer predicted by the model) and the *tLabel* (the ground truth provided by the user on whether the answer to the question is a lie or not). This table is the core table of the database as it connects other tables in our system. *sessionID* and *questionID* are the foreign keys from "Session" table and "Question" table. Attribute *sessionQuesID* also acts as a foreign key which appears in "Expression" table and "Gaze" table (discussed later). This helps retrieve each user's expression data for each question to train our model.

LogEvent: For each session, this table (Figure 7) keeps a record of the timestamps of the individual activities within the session. Specifically, it stores the start time and the end time for each test as well as for each question, and this helps assign a *sessionQuesID* for each record in Expression table and Gaze table.

Expression: This table (see Figure 8) keeps track of facial expression data which is captured every 50 milliseconds by affdex API. Each facial expression data contains 34 attributes which can be used to analyze the user emotion. Apart from that, *sessionQuestionID (assigned by comparing the expression timestamp with timestamps in LogEvent table)* will also be included in each tuple so that different tables can be connected for further analysis.

Gaze: This table (see Figure 9) records users' eye gaze data captured every 60/70 millisecond by eyetracker API. It tracks coordinates (*x*, *y*) of gaze and *confidence* of the result. Similarly, this table also contains *sessionQuestionID* and *timestamp*.

2.3 Modelling

The modelling part can be separated in 3 stages: data cleaning and preprocessing of raw data, training a machine learning algorithm and predicting the question in which a user is lying.

2.3.1 Data cleaning stage

As explained before, all the data provided by the affdex and eyetracker API are grouped with the same question and session id. The affdex API gives around 20 rows of data per second (timestamp) and more than 30 gesture variables such as joy, anger, surprise, smile, chin raise, lip press, etc., which means that for each question, that lasts for 10 seconds, we have a 200 x 30 matrix of data. This is why a cleaning process and dimensionality reduction techniques must be used.

When we first looked at the raw data, we checked the relevance of each feature of the data to our application purpose. We then manually decided to filter out irrelevant data such as age, ethnicity, gender, glasses etc. Also, the data collected by the eyetracker API has shown to have very low accuracy, thus for this first version of the application we have excluded it from the modelling. Here we ended up with 30 features for each row (timestamp) of data.

Afterwards, we reduced the number of rows of data by dividing them into 10 groups (each group tries to represent each second), and take the mean for each of the features or columns. With this, we ended up with 10 rows of data, each one with 30 features.

Lastly, we reshape all the data to 1 row and 150 features - similar to the data manipulation for image recognition - which is then loaded into our models as the X variables.

2.3.2 Training stage

After cleaning the data, we used it to train our model. Because of the limited data we had for training at this stage, we chose to construct four machine learning algorithms and make them compete in accuracy:

1. Support Vector Machine Classifiers
2. Random Forest Classifiers with Adaboost
3. Logistic regression with Adaboost
4. Decision Tree Classifier with Adaboost

Each model will use the 150 features (X) to determine whether the user lied or not in that question ($Y \in \{0,1\}$). The training stage will run once per day, using all the labeled data available in the database. During this stage, all 4 models will be trained, and each one will have an accuracy score. At the end, the model with the best accuracy will be selected, and it will be saved using pickle as the best model of the day.

2.3.3 Predicting stage

When a user is using the lie detector, the application will send only the data of the current session (data associated with the 3 questions presented to the user) to a predicting function. This function will, as in the training stage, clean and preprocess the data to prepare it to the model. Then, it will load the model saved with pickle (the one with highest accuracy), and use it to estimate the probability of each question being a question answered with a lie. Finally, the function will return the number of the question (1, 2 or 3) with the highest probability of being a lie. In case of more than one answer having the same highest probability, then there will be a random selection among them to return it to the user.

2.3.4 System limitations

One issue we have faced in our project is that we didn't have big enough data to train our models well before the due date. Therefore, we have included more modeling methods and have come up with our daily training model as some different model works different with different size of training data set. However, our all models we built do have shown to be capable to learn by themselves, thus we do have confidence that the accuracy will improve as the data size increase.

While many research has suggested the eye gaze is a good indicator of deception, we have faced issue in incorporating the data collected by the eyetracker API, as the accuracy was just too low. It has roughly been able to tell which direction the user is looking at (left/ right), but the data would be a mix of the user reading the question and gazing around as a subconscious reaction of lying. We eventually had to give up incorporating those data in our model regretfully. However, we do know that there are similar products with higher accuracy available in the market that are not free, and could definitely improve our application.

3 Project management and retrospective

From the start of the project, we have been using scrum methodology in an agile style by doing bi-weekly meetings in order to define milestones for the project and individual tasks to be completed. Starting from thinking about ideas to be developed, to research of papers, possible software components and APIs to get data for the goal of lie detection, definition of questions to provoke emotional reactions, development of web interface and documentation writing.

Although we separated the development of the project in 3 main tasks, we discussed everything we did with everyone, because of two main reasons. First, it is one project. We had to communicate what we were doing so that, at the end, the connection of all parts would be done smoothly. And secondly, we all helped each other with both technical knowledge as well as ideas of how to develop each one parts.

The main tasks of this project and the members responsible of them were:

a) Brainstorming ideas: All members
b) Front end design and user interface: Diego, Jorge
c) Data cleaning and modelling: Leila, Jenny
d) Database structuring and storage: Weisi, Xing
e) Report and presentation: All members

4 References

Porter, Stephen and Brinke, Leanne ten. “Reading Between the Lies. Identifying Concealed and Falsified Emotions in Universal Facial Expressions”. May 1, 2008. https://pdfs.semanticscholar.org/4e21/873c0af75a24d3ebf6359507b68f399831df.pdf [Accessed on 14 Nov 2017]

Reed, Lawrence Ian. Zeglenb, Katharine N. and Schmidtc Karen L. “Facial expressions as honest signals of cooperative intent in a one-shot anonymous Prisoner's Dilemma game”. *Evolution and Human Behavior*, Volume 33, Issue 3, May 2012. https://www.sciencedirect.com/science/article/pii/S1090513811001012 [Accessed on 4 Nov 2017]

Ordóñez, Francisco Javier and Roggen, Daniel. “Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition”. January 18, 2016. http://www.mdpi.com/1424-8220/16/1/115 [Accessed on 2 Dec 2017]

Thomas O. Meservy, Matthew L. Jensen, John Kruse, Judee K. Burgoon, and Jay F. Nunamaker Jr Meservy, Thomas O. Jensen, Matthew L. Kruse, John. Burgoon, Judee K. and Nunamaker Jr, Jay F. “Deception Detection through Automatic, Unobtrusive Analysis of Nonverbal Behavior”. https://www.researchgate.net/publication/3454301_Deception_Detection_through_Automatic_Unobtrusive_Analysis_of_Nonverbal_Behavior [Accessed on 19 Nov 2017]

Frank, Mark G. & Feeley, Thomas Hugh. “To Catch a Liar: Challenges for Research in Lie Detection Training” *Journal of Applied Communication Research*, 31:1, 58-75, 10 Jan 2011. http://www.tandfonline.com/doi/abs/10.1080/00909880305377 [Accessed on 20 Nov 2017]