Ecole Polytechnique Fédérale de Lausanne

LauzHack

DeCov - getting out of isolation safely

Authors :
Prune August, Lola Danhaive,
Jessica Gumowski, Simeon
Vareilles, Gabin Leroy, Lilian
Rouzaire

Version 0.1 -
6 avril 2020

1 Motivations

People around the world are holding their breaths, staying at home practicing social distancing to reduce the spread of covid-19. While life as we know it seems to have stopped, it is important to think of how to make it start again when the moment will be right. The challenge governments are facing is how to get out of isolation in a structured way, without risking another crisis. Inspired, amongst others, by the way South Corea tackled the pandemic, our team wants to provide authorities a tool that would make a safe "de-confinement" possible : a smartphone app called DeCov.

For each user, DeCov establishes a social contamination index, computed using several risk-factors. If the user has a high index, going out of isolation presents a danger to others. This could be the case if the user was recently tested positive or is closely related to a covid-19 positive person. A low index however would mean that getting out of isolation is safe. This could be the case if they have been staying in isolation for three weeks without showing any symptoms or if they are covid-19 negative. With time as well as the increasing number of immunized citizens, the risk index will progressively decrease for multiple people, thus assuring a smart controlled way out of isolation.

DeCov aims to be a non-centralized system that will mainly use Bluetooth in order to identify risky social interactions. As a result, each user's index will be affected by the indexes of the other users they cross, all without having access to other data. It is important to note that the user will have to provide little personal information : if they agree to, basic health aspects concerning their covid-19 history will be registered by the app. Since the index will be updated in real time, the data used for the computation will be deleted after that is done.

Additionally, a map representing the ratio of infected cases in different countries, regions and even cities would be made available in order for people to not only see the evolution of the pandemic but also to evaluate their own risk to go out, consequently being able to adapt their behaviour.

DeCov takes into account individual factors and thus provides a tailor-made solution to a very global problem. All this while being easy to implement, to manipulate, and using very few personal data. Local authorities could for example set for a particular risk-level under which people are allowed to go outside - citizens not respecting this index would have to stay at home until it is safe again for them to go out.
Different case scenarios :

— A user met a high index person a week ago (interaction identified by the app), but does not currently have any symptoms. The user's index is rather high, suggesting they should stay isolated. Additionally, if that user is to develop symptoms, they could become a priority for covid-19 testing compared to other, low risk index people. The index will then be changed according to the test results.

— A user is currently infected and has taken a test. This information is then indicated in the application, which will update to a very high risk and suggest that the user must remain isolated.

All the success of this app will be based on the common awareness that personal data is shared in a collective concern, but will obviously remain confidential (only GPS/Bluetooth and few health status details concerning COVID-19 would be required). The information will be encrypted and

not be shared to other users.
Let's introduce now the paths to construct how would, automatically, change the index.

2 Constructing the Algorithm

2.1 Introduction

In sight of simulating the variation of each individual's index, one must define the probability $P(x_1, ..., x_n)$ (depending on multiple variables) of getting contaminated while crossing a potentially infected person. Then it becomes necessary to simulate a Machine Learning Algorithm that would be able to correctly define these probability of transmission based on data received from the app as well as from external research on the virus. The probability must also be constantly updated, using statistical data from the app, in order to limit and overall avoid the propagation of errors. This implies the updated of the index would be constructed as the product between the index of the person the user has come in contact with and the probability of transmission. Lacking the skills in Machine Learning to code this algorithm, we will focus on explaining the functioning of the algorithm which uses all of the available data to simulate the evolution of the user's index in real time. This machine Learning Algorithm will be called $\mathcal{M}$.

2.2 Conception of the algorithm

2.2.1 Definition of the parameters, specific to each user

Let N be the size of a group of individuals. Let U be a specific user among this group, with a deCov index $i_U \in [1, 5]$. The precision of the parameter p depends on multiple factors, as well as certain choices made by the user. For more information about this point, see section 2.4.3.

— Let p be the parameter indicating the position in space of the user at the moment of contact with another individual.

— Let $\{f_1, ..., f_c\}$, $c \in \mathbb{N}$, be a set of variables representing the official documents provided by the user. In a first phase of the app, the only such document will be the document confirming a positive result to the Covid-19 test, meaning the app can only receive one document (the one mentioned here). In future versions of the app, one can eventually take into consideratiion more documents regarding other aspects of the user's health, medical history, ect.

— Let $\{a_1, ...a_k\}$, $k \in \mathbb{N}$, be other parameters concerning the user's current situation, that will also have an important impact on the calculation of his index.

2.2.2 INPUT

In addition to the parameters relative to the user (which have been presented in the previous section 2.2.1), the app receives an input of parameters relative to its other users. In fact, when two users find themselves at a small enough distance from eachother, the app gathers and saves the following :

— The index $i_{U,j} \in [1, 5]$ of the individual j met by the user U ($j \in [1, N]$)

— The date of the interaction d

- t_c : the time of contact between the two indivuals, meaning the amount of time during which the user and individual j where at a distance smaller than a reference distance d_0 (yet to be defined)
- Identification of the individual j

The daily indexes od the people that the user met, along with their time of contact, are stored in a $5 \times b$ table named `tab`. The 5 colums represent the following : the index, the time of contact t_c, the date d, the identifier of the individual, and the location of the user p. On the other hand, b represents the number of people that the user had contact with during the day. Another table `tab2` will allow to store the tables `tab` from the past n days (its size will hence be $5 \times \ b \times n$).
Let `test` be a boolean variable. If a document f_p regarding the results of a Covid-19 test has been added by the user, `test=true`, and if not, `test=false`. A variable `iftrue` is associated to each document, indicating if the document represents a postive result or a negative one. This means that if the document is a positive test result, `iftrue=true`. If i is your personal index, let i_j be the index associated to the j^{th} individual met by the user.
Let a be a variable that is updated depending on the location where the user is at the time of the interaction with another individual. If in that zone there is a high, medium, or low density of people, then $a = 2$, $a = 1$, and $a = 0$ respectively.

In addition to the parameters that are related to the user, presented in section 2.2.1, the app has also inputs related to the other users. Indeed, when two different persons meet at a short distance, the app registers,

- the index $i_{U,j} \in [1, 5]$ from the j^{th} person met($j \in [1, N]$)
- the meeting date d
- t_c defined as the contact time between the two persons, *it* the time during they have been in contact under a critical distance.
- The identification number of the j^{th} person.

The indexes of the persons we met and the time of contact is registered every day and stored in a table which has a size $5 \times b$ named `tab`. This table has a column related to the index, a column related to the time of contact, an other related to the meeting date, a column for the identification number and a final one related to the position of the meeting p. b represents the number of persons met in a day. Table `tab2` stores the different tables `tab` for the n^{th} last days (its size is $5 \times b \times n$).

Let's define a Boolean variable `test`. If a test medical document regardin COVID-19 is present, then `test=true`, otherwise `test=false`. At every time, we associate also the variable `iftrue`, which, if the document indicates a positiv test to COVID-19, is turned at `true`.

2.2.3 Body of the algorithm for a specific user

As stated in the previous section, we consider that two users enter into contact when there distance is less that l_s, that we must define, and which we consider as fixed for the moment. Indeed for the moment the goal of the app is to informed the user on his situation regarding the virus, it does not yet advice the user to approach somebody or not. It is a bluetooth system that inform to the algorithm that the distance between the the two users is less than l_s. Each time to users enter into contact, the algorithm does the following steps :

- `a` is assigned to its value
- memorisation of the indice $i_{U,j}$ of the other user j

— memorisation of the date d

— depending on the parameters $p, f_1, ..., f_c, t_c, a_1, ..., a_k$, an algorithm $\mathcal{A}$ updates the index i_U : see section 2.3.

All these information are memorised in `tab` and at the end of the day, they are added to `tab2`. We define the end of the day the moment when the clock exceed midnight as usual. Each time that the user U add a document of certification which relates to a test to the coronavirus (a document that indicate whether or not U is infected by the virus), then :

— if `test=true`, then the parameter $f_k, k \in \{1, ..., c\}$ is called

— if the variable f_i indicates a positive result to the test, then the index takes the value 5,and 1 if not.

In order to keep the user informed in the more precise way, the app uptade constantly the indice every r seconde. The parameter r must be choosen. Every r secondes, a retroactive algorithm is called in order to uptade the index i_U (see section 2.3).

2.3 Algorithms used to update the index

When users enter in contact, an algorithm $\mathcal{A}$ is called, and the only thing that it does is to call the machine learning algorithm $\mathcal{M}$ that gives a probability $P(p, t_c, a_1, ..., a_k)$ of transmission which does not depend on $i_{U,j}$, on j's health status. $\mathcal{A}$ uses the probability P to update the indices i_U as a function of $i_{U,j}$:
If $i_{U,j} \times P(p, t_c, a_1, ..., a_k) > i_U$,

$$i_U = P(p, t_c, a_1, ..., a_k) \times i_{U,j} \tag{1}$$

For the moment, the index of U only change when he met an other user. But, we know that at the moment of the meeting, the index $i_{U,j}$ of j may not reflect the exact health status at this moment so we need to take into account the change of indices of j before and after the date d of the meeting ; that is the goal of the retroactive-algorithm. Every r seconds, it does the following :

— A loop on all the identification number that are present in `tab2`

— for each identification number of the users $j \in [1, N]$, it evaluates if there is a change in the indexes of the user j around the date d due to the fact that j had an attestation paper $f_l, l \in \{1, ..., c\}$ which indicate that he is positive to the virus.

— if there is a such a change, then, the algorithm re-compute the index i_U at d by using the algorithm $\mathcal{A}$ with the parameter $p, t_c, a_1, ..., a_k$ of the date d and the new index $i_{U,j}$. The retroactive algorithm multiplies the all result given by $\mathcal{A}$ by an other term that depends on the time t_e that elapsed from d to the date of the change of index of U_j and on the difference of the change in index :

$$i_{U_{\text{new}}} = P(p, t_c, a_1, ..., a_k) \times i_{U,j} \times f(t_e) \tag{2}$$

The retroactive algorithm also update the current indice i_U by multiplying it by a function g of the time t_l that elapsed since d (this function describe the probability of transmission as function of time) and by a term that depends on the difference $\mid i_U - i_{U_{\text{new}}} \mid$

Obviously there are many factors that may imply a change of index of j after the meeting at date d ; for the moment, we don't take them into account due to a lack of time but we are fully conscious that these other factors will have to be studied.

2.4 Improvements and further considerations

2.4.1 Official documents and certificates

Some COVID19-related documents can be entered in the app, potentially with the agreement of health authorities.
Indeed, it is possible to enter a medical test result in the app. If the agreement of health authorities is needed, this could be done with a code provided by the medical center to enter in the app to ensure the validity of the test result, and consequently to adjust the index.

2.4.2 Deep Learning

In order that the algorithm automatically adjusts the contamination probability, it would be wise to compare the user index before and after the time the document was submitted on the app (see 2.4.1). In fact, if the index of the user is too low (or too high) compared to the real index, the algorithm could detect irregularities and change the way it takes into account test results in the process of re-evaluating the user index after a test result.

2.4.3 Cough detection and alerts

Acceleremoters or microphones on devices can be used to detect cough with a acceptable precision. This method is quite invasive, but could help detecting a chronic cough, or a punctual cough when a user meets another user (however in this case, it is hard to draw conclusions about the virus transmission, because bluetooth can only compute the distance between users, and not their position, if they wear masks or not, etc.).
Presence or absence of cough will not directly impact the user index, but could help refining the algorithm.

3 Map

The interactive map available on the app has to goal of showing the evolution of the virus through the world. More precisely, the maps show the prevalence of the virus, meaning the percentage of the population in a given area that has been tested positive to the virus. This way, the users can easily access information, that is usually found on multiple different sites or in long lists of data, all on one app. Showing this type of data is more useful than simply the total number of cases because what is important is the density, not the number. We created a prototype of the map for Italy, to show what it would look like. This is visible on video created. Furthermore, we coded this interactive map for Italy. Of course, the code has to be polished and refined for the final result, but it is present in the submission in order to show that our project is realisable. In the following paragraph we will detail the coding.

The map is interactive, it means you can click on a given region and this opens a more detailed map of the desired region. the data in percentages is colored according to a color code (red for the main map, blue for the sub-map) which goes from the most transparent to the most opaque. To display this map we used the SDL2 graphic library and coded in C ++. The card is a photo. The image is divided into a grid and it is thanks to this that we can group the regions and recognize the clicked place. In order to facilitate the grouping of the regions, it suffices to "prepare" the map by telling it which coordinate corresponds to which region simply by clicking on it and the program will write the coordinates in a .txt file. In a second time it can then read this file. Once the cards are prepared, the program is provided with .txt files with the percentage values and it will read them and translate them in color code on the cards.

4 Data storage

In order to keep track of all users' crossings identified via bluetooth, we need to store the data of the crossings. As the app is based on the anonymity of the data collected and the reduction of the personal data, we need to use identifiers for users, and store little amount of data at each crossing : both identifiers, the indexes of both users at that time, and the probability of transmission computed by the algorithm.
The aim is to compute all probabilities as soon as possible, so that the central database does not keep any explicit personal information : all the information about the users' health is represented through the index. However the result of covid-19 tests have to be stored alone and cannot be represented through the index, to be able to make a difference between a user tested positive and a user that has a high index because of many risky behaviors. This information will be deleted when not needed anymore.
Sensitive data such as the indexes or the proof of a covid-19 are also encrypted.

5 Design of the app

DeCov is meant to be used by every member of our society, therefore its design was a very important part of our work. The goal is to make it clear and easy to manipulate. To this end, only three main sections were implemented : *My Profile, My Index* and the *Corona-Map* (see paragraph 4). These sections are represented on the main bar at the bottom of the screen, and accessible by simply tapping on the respective icon. In *My Index*, the coronavirus icon in the middle of the screen will change colours from green to black depending on the index level, respectively 1-5. A simple tap on the virus will lead to a daily update page, that gives some informations about the current state of the user's index, how it will most likely evolve and some adapted advice for every situation. Keeping recurring main colours (peach and salmon) as well as using a refined design makes DeCov a soothing app, which was very important for us. In fact, a pleasant aesthetic helps to bring a sense of comfort in such uncertain times.