Protection against Data Profiling by Adversarial Cloud Applications

Master Thesis Report

École Polytechnique Fédérale de Lausanne
School of Computer and Communication Sciences
Distributed Information Systems Laboratory
Cheng-Hsiang Chiu

Supervised by:
Prof. Karl Aberer
Mr. Hamza Harkous
Lausanne, EPFL, 2016

Acknowledgements

I want to show my deepest gratitude to my families. With their fully beloved supports, I finally can finish the thesis. I would also want to appreciate the assistances from my thesis supervisor, Mr. Hamza Harkous, who helps me a lot during the whole semester. Finally, I am grateful to the friends I meet in EPFL. Their encouragements accompany me through the studies at EPFL.

Lausanne, 14 January 2016 C.H. Chiu

Abstract

The goal of this project is to design and develop protection mechanisms for protecting users' data on personal cloud-storage providers (CSPs) against data analysis by 3rd Party Apps. As we have shown in a previous work, users are highly alerted when they know what insights can extracted about them from their data on Google Drive. This highlights the importance of such insights and the need for mechanisms for protecting the data from unsolicited 3rd parties that get unnecessary access to these insights. Such insights are typically extracted by using machine learning and natural language processing techniques, such as topic modelling, entity recognition, sentiment analysis, face clustering, etc.
One way to protect this data is to encrypt it in the first place before sending it to the CSP. However, this carries the disadvantage that users are no more able to use 3rd party services with protected data. Another way is for users to not install apps that require extra permissions. However, this is not always possible as there might not be an app that requests the minimal permissions and still gets the job done for the user. Hence, we propose a new mechanism that works by adding dummy files to the user's account in order to result in dummy insights generated instead of true insights. This way apps could still work, but when they analyse the whole cloud storage account of the user, they will not be able to generate a profile of the user that can be used in providing ads, spying, or for other purposes.

Key words: Privacy, Adversary Machine Learning, Cloud Spaces

Zusammenfassung

Das Ziel dieses Projektes ist es, entwerfen und entwickeln Schutzmechanismen für den Schutz der Daten der Nutzer auf persönliche Cloud-Speicheranbieter (CSPs) gegen Datenanalyse von 3rd Party Apps. Wie wir in einem früheren Arbeiten gezeigt, werden die Benutzer sehr alarmiert, wenn sie wissen, was Erkenntnisse über sie aus ihren Daten auf Google-Laufwerk extrahiert. Dies unterstreicht die Bedeutung solcher Erkenntnisse und die Notwendigkeit von Mechanismen für den Schutz der Daten vor unerwünschten 3. Parteien, die unnötigen Zugang zu diesen Einsichten zu erhalten. Solche Erkenntnisse sind in der Regel mit Hilfe des maschinellen Lernens und der Verarbeitung natürlicher Sprache Techniken, wie Thema Modellierung, Entity Recognition, Sentiment-Analyse, Gesicht Clustering usw. extrahiert
Ein Weg, um diese Daten zu schützen, ist es in erster Linie zu verschlüsseln, bevor dem LSP zu senden. Allerdings führt dies den Nachteil, dass Benutzer nicht mehr in der Lage, 3rd-Party-Services mit geschützten Daten zu verwenden. Ein anderer Weg ist, damit Benutzer Anwendungen, die zusätzliche Berechtigungen erforderlich installieren. Dies ist jedoch nicht immer möglich, da es möglicherweise nicht eine App, die die minimalen Berechtigungen anfordert, und trotzdem wird die Arbeit für den Benutzer durchgeführt werden. Daher schlagen wir vor, einen neuen Mechanismus, der durch Zugabe von Dummy-Dateien auf das Konto des Benutzers, um in Dummy Erkenntnisse statt der wahren Erkenntnisse generiert führen funktioniert. Auf diese Weise apps könnte noch funktionieren, aber wenn sie die gesamte Cloud-Storage-Konto des Benutzers zu analysieren, werden sie nicht in der Lage, ein Profil des Benutzers, die bei der Bereitstellung von Anzeigen, Spionage, oder für andere Zwecke verwendet werden können, zu generieren.

Stichwörter: Datenschutz, Adversarial Machine Learning, Wolke Spaces

Résumé

Le but de ce projet est de concevoir et développer des mécanismes de protection pour protéger les données des utilisateurs sur les fournisseurs de cloud de stockage personnels (DSP) contre l'analyse des données par les applications 3ème Partie. Comme nous l'avons montré dans un précédent travail, les utilisateurs sont très avertis quand ils savent ce idées peuvent extraite à leur sujet de leurs données sur Google Drive. Cela souligne l'importance de ces points de vue et la nécessité de mécanismes pour protéger les données de 3e parties non sollicités qui obtiennent un accès inutile de ces idées. Ces idées sont généralement extraites en utilisant l'apprentissage de la machine et des techniques naturelles de traitement du langage, comme sujet la modélisation, la reconnaissance de l'entité, l'analyse des sentiments, le regroupement du visage, etc.

Une façon de protéger ces données est de chiffrer en premier lieu avant de l'envoyer à la CSP. Cependant, cela comporte l'inconvénient que les utilisateurs ne sont plus en mesure d'utiliser les services 3ème partie avec des données protégées. Une autre façon est destiné aux utilisateurs de ne pas installer les applications qui nécessitent des autorisations supplémentaires. Cependant, ce ne sont pas toujours possible car il pourrait ne pas être une application qui demande les autorisations minimales et obtient toujours le travail pour l'utilisateur. Par conséquent, nous proposons un nouveau mécanisme qui fonctionne en ajoutant des fichiers factices pour le compte de l'utilisateur dans le but de dégager des aperçus fictives générées au lieu de véritables perspectives. De cette façon, les applications peuvent encore travailler, mais quand ils analysent l'intégralité du compte de stockage en nuage de l'utilisateur, ils ne seront pas en mesure de générer un profil de l'utilisateur qui peut être utilisé dans la fourniture de publicités, d'espionnage, ou à d'autres fins.

Mots clefs : Confidentialité, contradictoire de l'apprentissage machine, Espaces Nuage

Contents

List of Figures

List of Tables

1 Introduction

In this chapter, a brief introduction to the problem we mange to solve is presented. Besides, two feasible solutions are discussed in this chapter.

1.1 What is the problem

The increasing computing powers of central processing unit (CPU), the surging storage spaces and growing network bandwidth contribute to a solid hardware infrastructure for the developments of cloud computing. Technology companies, such as Microsoft, Google, Amazon, Yahoo, Facebook, Apple and so on, then introduce loads of cloud services which include enormous storage spaces, email application, document editor and calendar. Almost all works people used to do on the personal computers could be realized via these cloud services. Gradually, people start to store a huge amount of data on the servers and cannot get rid of the convenience they bring about. Furthermore, theses technologies companies allow users to install third-party apps, which further diversify the functionalities of cloud services.
While installing the third-party apps, users notice that they are requested to authorize a list of permissions to apps. Figure 1.1 demonstrates a typical snapshot of requested permissions which include the ability to access and manage the files in users' cloud storage spaces. "Drive Files to Dropbox" is an app which simply allows users to transfer files from Google Drive to Dropbox. For such a simple service, it is obvious that "Drive Files to Dropbox" is able to access more data than it actually needs via the permission "View and manage the files in your Google Drive." Although the act that demands unnecessary permissions does not make the app a malicious software, consumers cannot stop but begin to worry about the leakage of privacy for the first time.
In the technical report [34], Mr. Hamza Harkous states that over two thirds of the top one hundred popular apps require more data than they truly need. Regardless of requesting unnecessary permissions unintentionally or purposely, with those data in hand apps could profile users and get their precise insights, which leads the usage of third-party apps into a potentially serious problem that personal privacy are at risk right after even installing an easy and lightweight app. Although Google Play store claims a constant examination is taken place

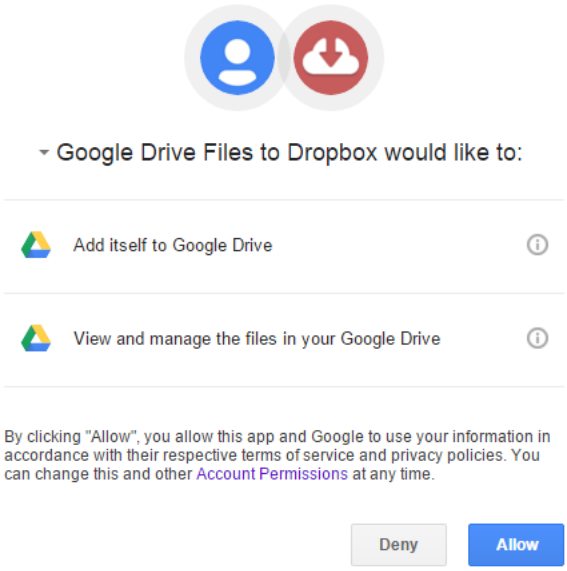


Figure 1.1 – Snapshot of permission requests by 'Drive Files to Dropbox'.

to remove malicious apps, it cannot be overemphasized the importance of precautions. There are many ways that the leakage of privacy could happen. In the thesis, we manage to deal with the one arising from the behavior that third-party apps are capable of drawing profiles of users. The leakage of documents is not the scope of this work.

1.2 Viable methods

To prevent third-party apps from getting personalized insights, the easiest way is to encrypt all files. In this way, profiling users will definitely be impossible for third-party apps. However, consumers will not be able to enjoy the services. For example, users can neither read encrypted documents with PDF Reader app [19] nor in Google Drive. Therefore, it is obvious that the method overcorrects the problem and is certainly not the solution we seek for.
Another approach is to add some dummy files in users' cloud storage spaces as a confusion for third-party apps. The advantage of this technique is that users could enjoy the services of apps along with keep their privacy safe. The downside, however, is inserting fabricated files in users' cloud spaces, which costs disk spaces as well as perplexes users.
In view of the enormous storage spaces users could get [38], let alone most users will not use up the storage spaces. Besides, a few megabytes of dummy files would result in a good obfuscation of the information once the algorithm of generating the dummy files is well developed. Although both methods have its own advantages and disadvantages, we decide to adopt the second way.

1.3 Structure of the thesis

The thesis will be given in the following structure. In Chapter 2, an introduction to background knowledge and related works will be discussed. Considering most of documents stored in the cloud space are composed of textual files and image files. In Chapter 3, the proposed methodology for adding dummy texts will be presented while the algorithm of adding dummy images will be introduced in Chapter 4. After that, the experimental results will be given. And final part will be the conclusions and future work.

2 Background and Related Works

In this chapter, some related works will be given. Before that, the permissions third-party apps could request and what kind of personalized insights they could get will also be stated.

2.1 Permissions that apps request

The permissions that we authorize to third-party apps are the main reason why they can profile users and get users' personalized insights. Hence, before getting to know the proposed methodology, we have to realize the permissions and what kind of abilities third-party apps are capable of once these permissions are authorized to them by users. The permissions here are involved in Google Drive only and can be applied to Drobox and OneDrive as well [13].

Table 2.1 – Permissions in Google Drive

Permission
View the files in your Google Drive.
View and manage the files in your Google Drive.
View metadata for files in your Google Drive.
View and manage metadata of files in your Google Drive.
View and manage Google Drive files that you have opened or created with this app.
View your Google Drive apps.
Add itself to Google Drive.
View and manage its own configuration data in your Google Drive.

As we can see in Table 2.1, there are eight permissions that third-party apps can request. And only one permission, "View and manage Google Drive files that you have opened or created with this app", is operated on a per-file basis. This permission gives the apps the ability to access only the files on which users intend to utilize the service the apps provide. That is to say, "Drive Files to Dropbox", the app that we introduced in Chapter 1 could only access the files that users want to transfer from Google Drive to Dropbox. And the other files that users have no intention to move will not be able to be accessed by the app. But, the permission that "Drive Files to Dropbox" requests is "View and manage the files in your Google Drive" which is a full access and will allow the app to access not only the target files but also all the files in users' Google Drive.
Viewing and managing metadata of files may seem to be harmless to users. The truth is metadata reveals more information than users can imagine. For textual files, important metadata could include owner's name, editing time, and file type. And XMP, Exif and IPTC are critical metadata in image files. Third-party apps could know locations users visited, camera type, patterns of consumers' habits of using computers and so on via accessing metadata. Therefore, as long as it is a full access that third-party apps request, no matter data are from files or metadata, the extra data they could access will become the potential concerns for privacy leakage.

2.2 Related works

In this section, we discuss some related works. The first subsection is about a technical report which clearly defines the issue that third-party apps request a full access to users' files and three works that inspire us to come up with the proposed methodology.

2.2.1 What insights apps could get

In this subsection, we depict the personalized insights apps could get from users' files. The insights include information of users and users' friends, families, or colleagues either in texts or images. In the work, the file types we cover are textual files and image files since they are the most popular types people store in the storage spaces.
Most of the contents of this subsection are extracted from the technical report [34] finished by Mr. Hamza Harkous who is affiliated in Distributed Information Systems Laboratory (LSIR) at École polytechnique fédérale de Lausanne (EPFL). Mr. Harkous also constructed a webpage [20] to demonstrate the personalized insights that third-party apps might get during the process of profiling. They are

1. Sentiment
2. Top Collaborators
3. Shared Interests

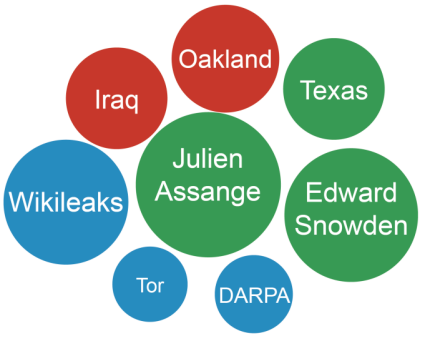


Figure 2.1 – ECT visual of a user, extracted from the report [34]

4. Faces with Context
5. Entities, Concepts, and Topics (ECT)
6. Faces on Map

This subsection only gives a simple introduction to the insights. The privacy leakages we try to tackle are ECT, Shared Interests, and Faces on Map. And they will be stated in more details in the subsequent chapters.

2.2.1.1 Entities, concepts, and topics (ECT)

We consider the top named entities, for example, Apple, coffee cognitive science and etc, which are presented in the textual files. The top named entities refer to the entities, which are either people or place, occur the most times in the text.
Concepts are extracted from the concept tags in users' documents, such as credit card, personal finance and so on. Concepts are the high-level abstraction. For example, an author might write the sentence "I like Kobe Bryant, Lebron James, Stephen Curry and Time Duncan." These four are named entities. But the concept of this sentence is basketball. So even the word "basketball" does not appear in the text. We could still get the concept of that text by analyzer which is discussed in the subsequent chapters.
Then we could get a ECT visual of a user in Figure 2.1, where the size of circle is proportional to the number of that topic appearing in the users' documents and different color corresponds to different entities. For example, the user has more documents about "Julien Assange" than "Texas."

2.2.1.2 Sentiment

We could get the user's positive, negative or neutral preference over the entities which are obtained in ECT.

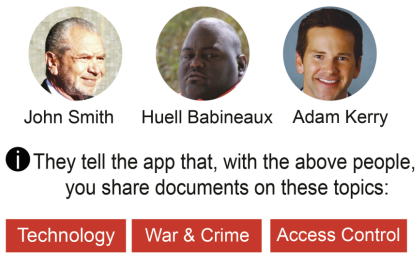


Figure 2.2 – SharedInterests visual of a user, extracted from the report [34]

2.2.2 Top collaborators

Users' top collaborators are defined as those people who are either the principals or the witnesses of shared files. Normally, top collaborators are users' colleagues, families or friends. Hence, any privacy leakage would involve more victims than users could imagine.

2.2.2.1 Shared interests

With this insight, third-party apps are able to know the mutual interests of users and the top collaborators. The visualization is demonstrated in Figure 2.2 where the users share documents related to "Technology", "War and Crime" and "Access Control" with the three top collaborators, "Jonh Smith", "Huell Babineaux" and "Adam Kerry." Therefore, it is really a bargain that third-party apps profile once and get insights for four people.

2.2.3 Faces with context

The above insights are associated mainly with users' textual files. The remaining two insights are based on users' image files. This insight indicates the most frequent faces as well as the concepts appearing in the user's images.

2.2.3.1 Faces on map

The last insight is about the geographical locations the users' photos would reveal in the Exif metadata. And Figure 2.3 is an example showing the locations users ever visited. This insight is usually obtained by top locations and top faces in all of the photos. Top locations mean that the location are constantly shown in photos and top faces represent the faces that show up in the photos the most times. Then, third-party apps could use the insights to recommend some stores near those locations to these people.

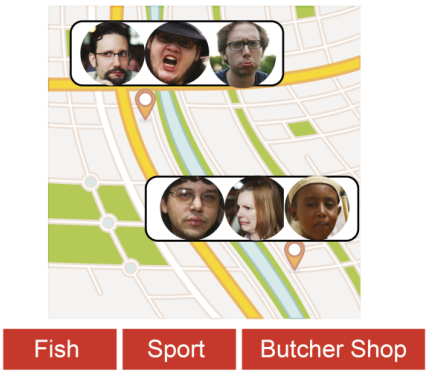


Figure 2.3 – FacesOnMap visual of a user, extracted from the report [34]

2.2.4 The works that inspire the proposed methodology

In this subsection, some related works about privacy protection are given. The works we search for are focused on information obfuscation instead of encryption.

2.2.4.1 TraceMeNot

TraceMeNot [33], developed by Mr. Daniel C. Howe and Mr. Helen Nissenbaum, is a browser extension which protects users from surveillance and profiling by search engines. The method is not to conceal or encrypt users' data but to make obfuscations so that search engines would have difficulties obtaining users' insights.

2.2.4.2 AdNauseam

AdNauseam [35, 5] was developed by the same research team who built TraceMeNot. AdNauseam is a browser extension that is able to protect users from revealing digital footprints to advertising networks. According to AdNauseam, users' ad-click pattern is also one of users' profiles. Hence what AdNauseam does is to automatically click the blocked advertisements on behalf of users, which turns out to be a visiting record on the ad networks databases. Therefore, users' pattern of surfing the Internet is obfuscated. The source code of AdNauseam is released on GitHub [9].

2.2.4.3 CacheCloak

CacheCloak [30, 31] was developed by Mr. Joseph Meyerowitz and Mr. Romit Roy Choudhury, which intends to anonymize mobile users' personal locations while firstly processing the location data in a trusted anonymizing server and sending back the processed data to untrusted location based services in a real-time basis. They claimed CacheCloak could protect users' location privacy without losing location accuracy or availability.

3 Methodology for Dummy Texts

In this chapter, the proposed method for adding dummy texts is presented. As mentioned in Section 1.2 and Section 2.2.4, the methodology in this work is to obfuscate information by adding dummy files rather than encrypting data. Dummy files include textual files and image files.

3.1 Three plausible text generators

Adding dummy texts increases the degree of difficulties for adversaries who aim to get true insights of cloud service users. To achieve the goal of confusing adversaries, the contents of dummy texts should not be a random combinations of characters, but quasi-human-written sentences. Besides, the contents of the fabricated texts are extracted from latest news with various taxonomies instead of some constant sources. The reason for that is to avoid adversaries from blacklisting certain source as being sources for dummy texts in case the sources are fixed. Therefore three plausible approaches fulfilling the needs are found and examined. They are Context-free grammar (CPG), Markov chain model (MCM) and neural networks. The following sections would be the introductions to the three techniques.

3.1.1 Context-free grammar

In natural language processing, Context-free grammar (CFG) [23] is a way to generate patterns of strings and is defined as follows:

Definition 1 *Context-free grammar*
A CFG, $\mathcal{G}$, is defined by a 4-tuple : $\mathcal{G} = (\mathcal{T}, \mathcal{N}, \mathcal{S}, \mathcal{R})$, where

- *$\mathcal{T}$ is a set of terminal symbols.*
- *$\mathcal{N}$ is a a disjoint from $\mathcal{T}$ and is referred to non-terminal symbols.*

- $\mathscr{S}$ *is a start symbol, which is one of the non-terminal symbols.*
- $\mathscr{R}$ *are rules/productions of the form* $x \rightarrow y$*, where* x *is a nonterminal and* y *is a sequence of terminals and nonterminals (may be empty).*

■

Therefore, sentences could be generated easily using several $\mathscr{G}$. An example is given to demonstrate the usage of CFG.

Example 1 *A simple example* $\mathscr{G}$ *is given.*

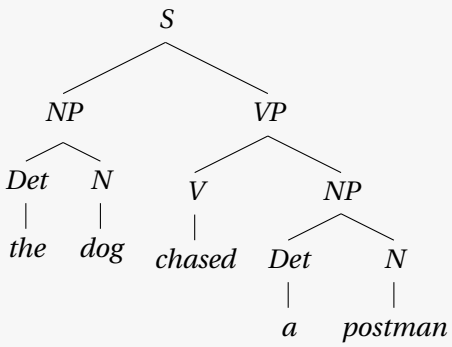
$$
\begin{aligned}
&\mathscr{T} = \{a, dog, chased, postman, the\} \\
&\mathscr{N} = \{S, NP, VP, N, V, Det\} \\
&\mathscr{S} = \{S\} \\
&\mathscr{R} : S \quad \rightarrow NP\ VP \\
&\qquad VP \quad \rightarrow N \\
&\qquad VP \quad \rightarrow V\ NP \\
&\qquad NP \quad \rightarrow Det\ N \\
&\qquad N \quad \rightarrow dog \\
&\qquad N \quad \rightarrow postman \\
&\qquad Det \quad \rightarrow the \\
&\qquad Det \quad \rightarrow a \\
&\qquad V \quad \rightarrow chased
\end{aligned}
$$

With the 4-tuple mentioned above, the sentece, "the dog chased a postman", could be derived by applying the sequences with the start symbol, S:

$$
\begin{aligned}
&S \\
&\xrightarrow{\mathscr{R}1} NP\ VP \\
&\xrightarrow{\mathscr{R}4} Det\ N\ VP \\
&\xrightarrow{\mathscr{R}7} the\ N\ VP \\
&\xrightarrow{\mathscr{R}5} the\ dog\ VP \\
&\xrightarrow{\mathscr{R}3} the\ dog\ V\ NP \\
&\xrightarrow{\mathscr{R}9} the\ dog\ chased\ NP \\
&\xrightarrow{\mathscr{R}4} the\ dog\ chased\ Det\ N \\
&\xrightarrow{\mathscr{R}8} the\ dog\ chased\ a\ N \\
&\xrightarrow{\mathscr{R}6} the\ dog\ chased\ a\ postman
\end{aligned}
$$

As we could see in the above, the sentence, "the dog chased a postman", is obtained by iteratively substituting the rules in the sequence : ($\mathscr{R}1, \mathscr{R}4, \mathscr{R}7, \mathscr{R}5, \mathscr{R}3, \mathscr{R}9, \mathscr{R}4, \mathscr{R}8, \mathscr{R}6$)*. One can also*

derive the sentence using a tree diagram, which is more straightforward to get the feeling of the derivation of the sentence.



From the above example, we realize that CFG is capable of generating human-written texts as long as the 4-tuple is well organized. ■

Several years ago, a group of researchers at MIT CSAIL developed a research paper generator, SCIgen, by using CFG to automatically generate computer science research papers[21]. And it took everyone by surprise that some of the fabricated research papers were accepted by academic conferences. Therefore, SCIgen, is undoubtedly a good candidate to generate dummy texts, even though it is no longer maintained by MIT CSAIL. The source code is available online[10].

After scrutinizing the source code of SCIgen, we found out there are over three-thousand lines of rules defined in advance in order to generate a convincing paper of one specified taxonomy. Since different types of texts require diverse sentence structures and vocabularies, twenty taxonomies imply twenty predefined $\mathscr{G}$. Nevertheless, the $\mathscr{G}$ has to be revised as long as the source corpus is updated from time to time in case adversaries blacklist the current source corpus. Although, CFG has been proven by MIT CSAIL to be a valid and powerful way via acceptances of research papers, its strengths are still compromised by lengthy rules and routine revisions.

3.1.2 Neural networks

Neural networks [22, 6] is an information paradigm that was inspired by biological nervous systems. Usually, the networks have an input layer, a hidden layer, and an output layer, as demonstrated in Figure 3.1. Each layer is composed of several neurons. The connections mean the excitations of the connected neurons. Every excitation is implemented with a weighted edge which is based on the excitatory degree of two connected neurons. And the output is equal to the linear combinations of the weighted inputs. If the network generates a bad result, then the weightings would be learned again. There are three strategies for learning, supervised

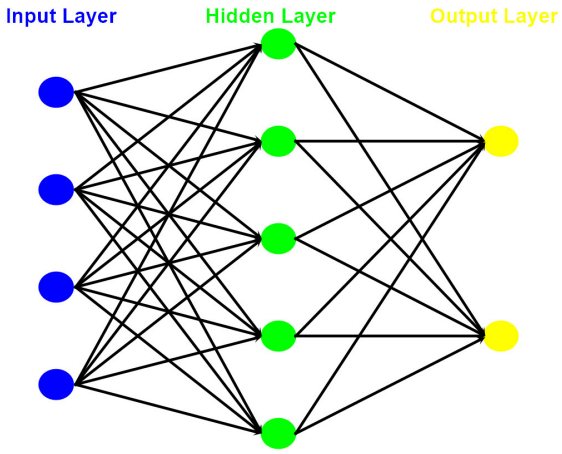


Figure 3.1 – Typical structure of neural networks.

learning, unsupervised learning, and reinforcement learning.
Back to text generator, Recurrent neural networks (RNN) [28] is a class in neural networks which is usually used to implement character or text generator. The feature of RNN is that the presence of the internal states, resulted from a directed cycle structure, contributes to the internal memory which could be used to process arbitrary sequences of inputs. Figure 3.2 shows a example structure of recurrent neural networks, in which a rectangle is a vector and red rectangle refers to input vectors, output vectors are in blue and green vectors keep the RNN's state. Before applying Neural networks to text generators, there is a simple example for a character generator which is shown as follows:

Example 2 *The example is extracted from the webpage [12].*
Suppose there is a list of vocabulary ['h', 'e', 'l', 'o']. And we would like to get a character "hello." By following the source code released on GitHub [12], the working procedure of this diagram is shown in Figure 3.3, where we could see the next letter of "hell" is "o" given letter "o" has the highest confidence, 2.2 depicted in color green in the blue rectangular, than the other letters. Each output neuron has four states because of four inputs.

■

The example demonstrates the way to generate a character. By studying the three papers [29, 32, 37], we can easily extend the character generator to a text generator. The advantage of neural networks text generator is that it could generate a text that is very close to a human-written one. However, the whole process is very time-consuming given the number of neurons and layers. In view of routine updates of source corpus, the processing time is the big concern of the neural networks text generator.

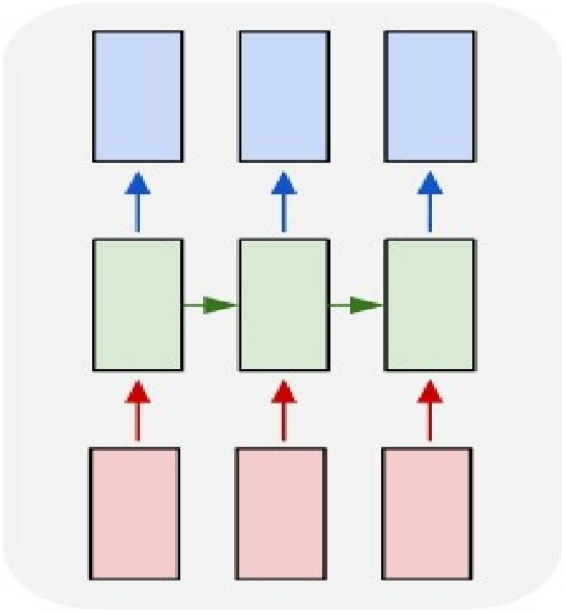


Figure 3.2 – Structure of recurrent neural networks.

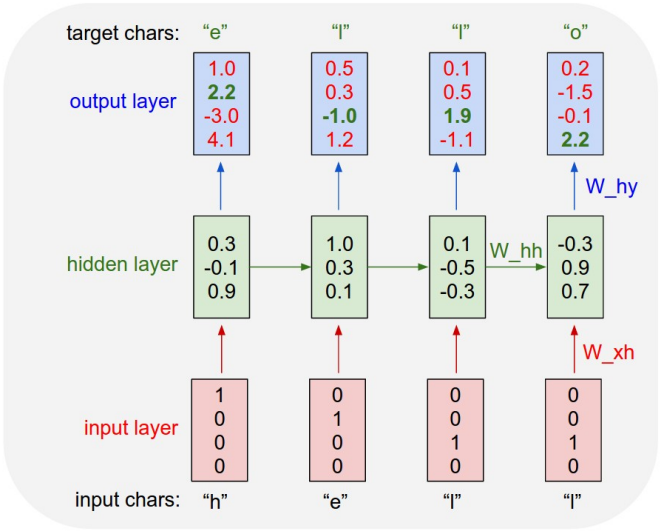


Figure 3.3 – Diagram of neural networks character generator, extracted from the webpage [36].

3.1.3 Markov chain model

Markov chain model is a sequence of states of a certain system in which a present state depends only on the previous state [27]. A detailed definition of Markov chain is given in the following.

Definition 2 *Markov chain*
A set of states, $\mathscr{S} = \{S_1, S_2, \ldots\}$. The process starts with an initial state and moves from one state to another state successively. Every move, called a step, is denoted with a transition probability. Assume the current state is S_i and the next state is S_j, then the transition probability of this step is equal to p_{ij}. And Markov property states that p_{ij} is independent of the state prior to S_i, which could be formalized in a mathematical expression.

$$Pr(S_{n+1} = s_{n+1} | S_1 = s_1, S_2 = s_2, \ldots, S_n = s_n) = Pr(S_{n+1} = s_{n+1} | S_n = s_n) = p_{ij} \tag{3.1}$$

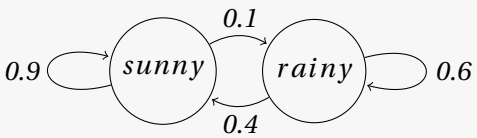
■

Next, we make a classical example, weather prediction, to briefly demonstrate Markov chain model.

Example 3 *Suppose there is a weather model in which a sunny day is 90% likely to be followed by another sunny day, and a rainy day is 60% likely to be followed by another rain day. The probability of having a rainy day the next day is 40% given a sunny day today and 10% vise versa. A transition matrix denoting the transition probability could be constructed.*

$$P = \begin{pmatrix} 0.9 & 0.1 \\ 0.4 & 0.6 \end{pmatrix}$$

The column of P is labelled 'sunny' and 'rainy' in order, so is the row. In addition to P, one could build a state diagram.



Assume the observed day is a sunny day with a vector $S_0 = (1 \quad 0)$, where 1 denoting the sunny entry and 0 presenting the rainy entry. Then the probability of next day could be calculated easily:

$$
\begin{aligned}
S_1 &= (1 \quad 0) * P = (0.9 \quad 0.1) \\
S_2 &= S_1 * P = (0.9 \quad 0.1) * P = (0.85 \quad 0.15) \\
&\vdots
\end{aligned}
$$

which shows the next day, S_1, is 90% likely to be a sunny day while 10% likely to be a rainy day. And S_2 is obtained by multiplying S_1 with P.

■

Now, we have a brief understanding of Markov chain model. Then, we could apply Markov chain model to generate a dummy text from a source corpus. And an example is a good way to demonstrate the procedure.

Example 4 *A text generator using Markov chain model*
Suppose the source corpus has some simple sentences,

Today is a sunny day.
Yesterday was a rainy day.
I like sunny days.
You like rainy days.
A sunny day is better than a rainy day.
An apply a day keeps doctors away.

Then, for each vocabulary, we calculate the occurring probability of its next vocabulary and have the following model,

Today	$\Longrightarrow$	*is (100%),*
is	$\Longrightarrow$	*a(50%), better(50%),*
a	$\Longrightarrow$	*sunny(25%), rainy(50%), day(25%),*
sunny	$\Longrightarrow$	*day.(100/3%), days.(100/3%), day(100/3%),*
day.	$\Longrightarrow$	*Yesterday(100/3%), I(100/3%), An(100/3%),*
Yesterday	$\Longrightarrow$	*was(100%),*
was	$\Longrightarrow$	*a(100%),*
rainy	$\Longrightarrow$	*day.(67%), days.(33%),*
I	$\Longrightarrow$	*like(100%),*
like	$\Longrightarrow$	*sunny(50%), rainy(50%),*
days.	$\Longrightarrow$	*You(50%), A(50%),*
You	$\Longrightarrow$	*like(100%),*
A	$\Longrightarrow$	*sunny(100%),*
day	$\Longrightarrow$	*is(50%), keeps(50%),*
better	$\Longrightarrow$	*than(100%),*
than	$\Longrightarrow$	*a(100%),*
An	$\Longrightarrow$	*apple(100%),*

$$
\begin{aligned}
\textit{apple} &\Longrightarrow \textit{a(100\%),}\\
\textit{keeps} &\Longrightarrow \textit{doctors(100\%),}\\
\textit{doctors} &\Longrightarrow \textit{away.(100\%),}\\
\textit{away.} &\Longrightarrow \textit{Today(100\%)}
\end{aligned}
$$

Let's explain the calculation of probability via the entry,

$$\textit{a} \Longrightarrow \textit{sunny(25\%), rainy(50\%), day(25\%)}$$

In the source corpus, vocabulary 'a' is followed by vocabulary 'sunny' once, 'rainy' twice and 'day' once. Hence, 'a sunny' occurs one out of four as encountering vocabulary 'a'. Next, we could generate a dummy sentence by randomly pick a vocabulary from the set, {Today, Yesterday, I, You, A, An}, where each element begins with a capitalized letter.
Assume the start vocabulary is 'An'. The next vocabulary can only be 'apple', which is followed only by 'a'. Until now, we have the generated sentence, 'An apple a'. The subsequent vocabulary has three options, 'sunny', 'rainy' and 'day'. Without loss of generality, we pick 'rainy' over the other two options because of its higher probability. Afterwards, 'day.' is chosen over 'days.' as well. In the end, the generated sentence becomes 'An apple a rainy day.'
The procedure of the generated sentence is as follows:

An
An apple
An apple a
An apple a rainy
An apple a rainy day.

■

The advantage of this kind of text generator is quickness while training a big source corpus. However, the disadvantage is that the fabricated sentence may not make any sense. To avoid generating meaningless sentences, the idea of n-gram is introduced to make a n-gram Markov chain model, which is presented in the following example.

Example 5 *A text generator using n-gram Markov chain model*
The Markov chain model, introduced in Example 4, is called unigram Markov chain model since the current state is dependent on it previous state only. N-gram Markov chain model, as the name suggests, refers to the fact that current state depends on the previous n states. Let's run bigram on the source corpus stated in Example 4 and would get the following model,

Today is	$\Longrightarrow$	*a (100%),*
is a	$\Longrightarrow$	*sunny (100%),*
a sunny	$\Longrightarrow$	*day. (100%),*
sunny day.	$\Longrightarrow$	*Yesterday (100%),*
day. Yesterday	$\Longrightarrow$	*was (100%),*
Yesterday was	$\Longrightarrow$	*a (100%),*
was a	$\Longrightarrow$	*rainy (100%),*
a rainy	$\Longrightarrow$	*day. (100%),*
rainy day.	$\Longrightarrow$	*I (100%),*
day. I	$\Longrightarrow$	*like (100%),*
I like	$\Longrightarrow$	*sunny (100%),*
like sunny	$\Longrightarrow$	*days. (100%),*
sunny days.	$\Longrightarrow$	*You (100%),*
days. You	$\Longrightarrow$	*like (100%),*
You like	$\Longrightarrow$	*rainy (100%),*
like rainy	$\Longrightarrow$	*days. (100%),*
rainy days.	$\Longrightarrow$	*A (100%),*
days. A	$\Longrightarrow$	*sunny (100%),*
A sunny	$\Longrightarrow$	*day (100%),*
sunny day	$\Longrightarrow$	*is (100%),*
day is	$\Longrightarrow$	*better (100%),*
is better	$\Longrightarrow$	*than (100%),*
better than	$\Longrightarrow$	*a (100%),*
than a	$\Longrightarrow$	*rainy (100%),*
a rainy	$\Longrightarrow$	*day. (100%),*
rainy day.	$\Longrightarrow$	*An (50%), I(50%),*
day. An	$\Longrightarrow$	*apple (100%),*
An apple	$\Longrightarrow$	*a (100%),*
apple a	$\Longrightarrow$	*day (100%),*
a day	$\Longrightarrow$	*keeps (100%),*
day keeps	$\Longrightarrow$	*doctors (100%),*
keeps doctors	$\Longrightarrow$	*away. (100%),*
doctors away.	$\Longrightarrow$	*Today (100%),*
away. Today	$\Longrightarrow$	*is (100%)*

As we can see, most entries are followed by one only vocabulary, which largely reduces the possibility of randomization. Assume the start bigram is 'An apple', then by following the model the final sentence could only be 'An apple a day keeps doctors away.' Then the fabricated sentences are more likely to be the original sentences as N increases, which can be good if we desire a meaningful sentence, but would be a disaster for our implementation. Since we want the dummy sentences to be as random as possible, the bigger N might lead to the situation that all

sentences are originated from a single news report and adversaries could easily recognize the dummy texts.

■

3.1.4 Comparison of three text generators

The introductions to the three text generators have been given above. And each one has its advantages and disadvantages. Now, we run a simple pros and crons analysis of the three text generators. The main comparisons are focused on the execution time, including the training time of source corpus and sentences generating time, as well as the degree of convenience when it comes to updating the source corpus on a daily basis. The analysis is shown in the following table.
Since the source corpus will be updated routinely and the execution time of the text generator is required to be as short as possible, we choose to apply Markov chain model in the text generator. In Table 3.4, although, the fabricated texts of Markov chain model is less like human-written sentences comparing to the other two method, the idea of N-gram is introduced in Markov chain model to increase the likelihood, as given in Example 5. Therefore, we choose N-gram Markov chain model as the text generator.

Table 3.4 – Pros/Crons of three text generators

	Pros	**Crons**
Context free grammar	– self define sentence structure	– every taxonomy needs unique rules
	– like human-written sentences	– updating rules routinely is required
	– execution time is little	
Neural networks	– like human-written sentences	– execution time is too much
	– easy to operate	– training is time-consuming
Markov chain model	– training a model is fast	– less like human-written sentences
	– updating source corpus is easy	

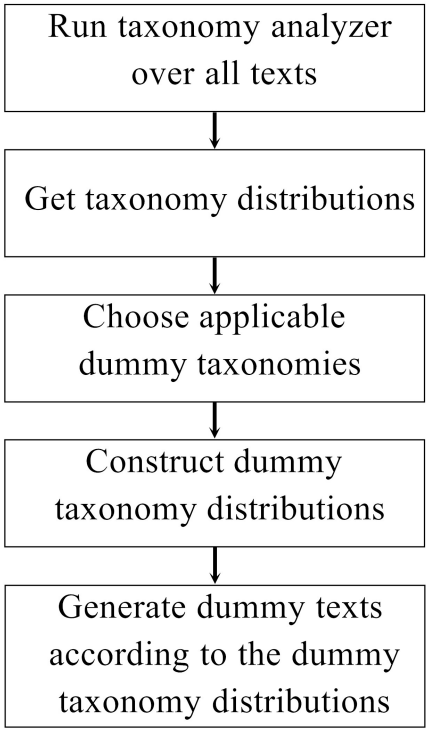


Figure 3.4 – Flowchart of generating dummy texts.

3.2 Our implementation to add dummy texts

In this section, the implementation of generating dummy texts is given. The flowchart is demonstrated in Figure 3.4. In the following subsections, each main function block will be explained with more details. Before introducing each block, the process to update source corpus is given firstly.
The idea behind this implementation is to generate dummy texts based on top taxonomies which refer to those taxonomies appearing the most times in all of the textual files. And we intend to add dummy texts as many as the real texts. Although the dummy texts are generated in the perspective of top taxonomies, we would show the obfuscation of texts still work for top concepts and top entities in the experiments.

3.2.1 Updating source corpus

The process of updating the source corpus dose not apply to every user but is performed every day. The reason is that the dummy texts are more convincing if the contents come from the latest news reports, which sheds an illusion to adversaries that the account of the personal cloud spaces is being accessed routinely. Furthermore, the process makes sure the source corpus is big enough to decrease the chance of generating two similar dummy texts of same

taxonomy and prevent adversaries from blacklisting our source corpus.
To gather daily news reports, we use AlchemyAPI [2] which, now part of IBM Watson Developer Cloud [15], is a popular cloud platform where software developers could utilize the advanced text analysis api [3] and computer vision api [4] to build useful applications.
While gathering news reports, developers choose interested taxonomy and publication date in the AlchemyAPI. The returned results are expressed in a JSON format, including title and url. Then, we capture the contents in the returned urls and store them in the database for references. And to improve the process of generating dummy texts, contents of same taxonomy are concatenated together and stored in a local file.
The reasons of gathering news reports with taxonomies rather than concepts or entities are that AlchemyAPI only allows users to gather news with taxonomies and it is not possible to list all the concepts or entities in the news reports.

3.2.2 Taxonomy analysis

In order to run the taxonomy analysis, we have to access users' files in Google Drive. In this website [20], registered users have authorized us the permission to access and manage all of their files in Google Drive. By taking advantage of alchemy language api [1], the taxonomy analyzer could analyze a text and output the taxonomies along with sentiments, concepts and entities of that text. And the outcomes of the taxonomy analyzer are stored in our database. Since the taxonomy analysis is a preprocessing step, we assume all registered users have been analyzed and what we do now is to query the database to get the outcome of a target Google Drive user, called G-User for convenience in the subsequent contents. The query to the database is taxonomy-oriented since the dummy texts are generated using specified taxonomies and is defined in the following:

```
db.find
({
    'fileKind': 'text',
    'textAnalysisResults.taxonomies': {$get: []}
})
```

After all analyzed results are fetched from the database, we could visualize the outcomes in a histogram. Let's make an example to show the visualization to get an idea what top taxonomies actually mean. Figure 3.5 is an example demonstrating the histogram in which 'Sports' is the top one taxonomy because it appears in fifty eight texts owned by target G-User. Until now, we have a taxonomy distributions of real texts owned by the target G-User and this real taxonomy distributions are demonstrated in the following data structure,

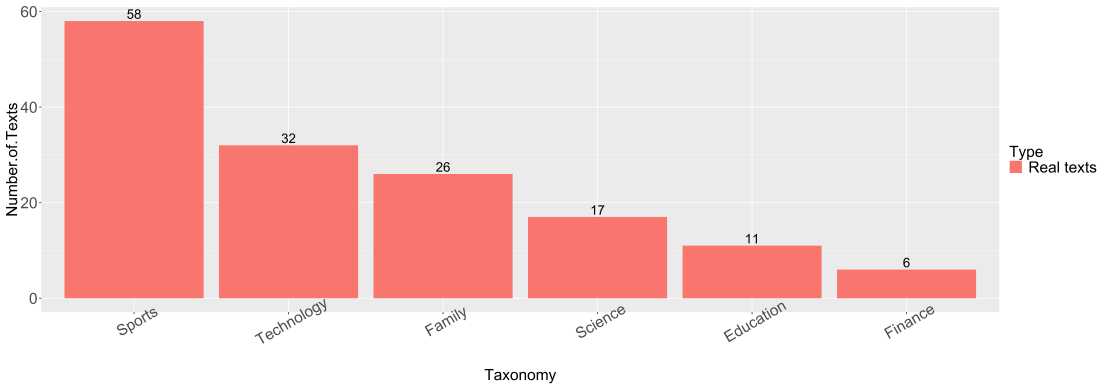


Figure 3.5 – Demonstration of taxonomy distributions of a G-User

```
[
    {taxonomy: 'Sports', count: 58}, {taxonomy: 'Technology', count: 32},
    {taxonomy: 'Family', count: 26}, {taxonomy: 'Science', count: 17},
    {taxonomy: 'Education', count: 11}, {taxnomy: 'Finance', count: 6}
]
```

3.2.3 List of dummy taxonomies

Dummy taxonomies are defined as the ones which do not appear in the list of real taxonomies obtained in the first part. Alchemy api provides a list of twenty three taxonomies, listed below. Hence, the dummy taxonomies could be obtained by discarding the real taxonomies from the list of available twenty three taxonomies.

```
[
    'Art and Entertainment', 'Automotive and Vehicles', 'Business and Industrial',
    'Careers', 'Education', 'Family and Parenting', 'Finance', 'Food and Drink',
    'Health and Fitness', 'Hobbies and Interests', 'Home and Garden',
    'Law, Government and Politics', 'News', 'Pets', 'Real Estate',
    'Religion and Spirituality', 'Science', 'Shopping', 'Society', 'Sports',
    'Style and Fashion', 'Technology and Computing', 'Travel'
]
```

The number of the real taxonomies, however, might be close to twenty three and would lead to a list of few dummy taxonomies. Hence, we pick the top ten taxonomies from the real taxonomies and remove these ten taxonomies from the twenty three taxonomies. The algorithm is stated as follows,

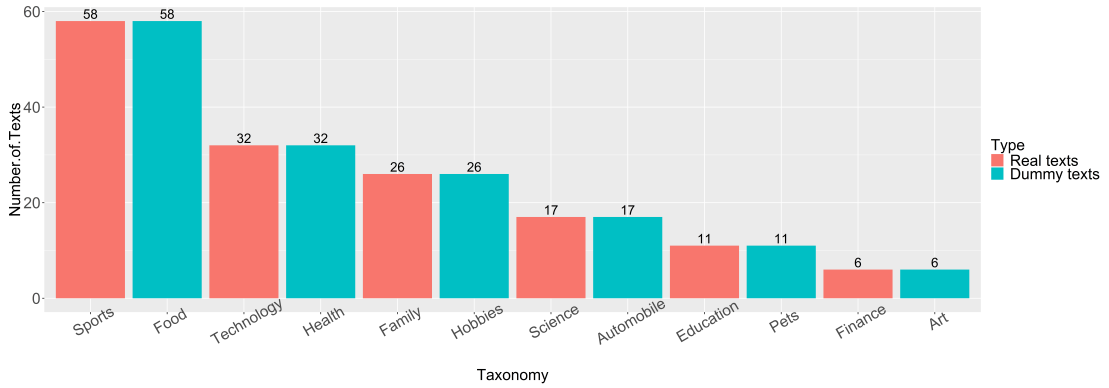


Figure 3.6 – Demonstration of post texts distribution of a G-User

1. if((the number of real_taxonomies) > 10)
   real_taxonomies = top 10 in real_taxonomies

2. dummy_taxonomies = all_taxonomies \ real_taxonomies

3.2.4 Dummy taxonomy distributions

Now, we have a list of dummy taxonomies. The next work is to construct a dummy taxonomy distributions. Since we want to add the same number of dummy texts as the real texts such that the example histogram after adding the dummy taxonomies would be look like the one in Figure 3.6 in which the dummy taxonomies are 'Food', 'Health', 'Hobbies', 'Automobile', 'Pets', and 'Art.' Combining with the number of dummy texts, a dummy taxonomy distributions would be constructed in the following,

```
[
    {taxonomy: 'Food', count: 58}, {taxonomy: 'Health', count: 32},
    {taxonomy: 'Hobbies', count: 26}, {taxonomy: 'Automobile', count: 17},
    {taxonomy: 'Pets', count: 11}, {taxonomy: 'Art', count: 6}
]
```

While implementing the methodology, we only consider top five taxonomies. Hence, we randomly pick five dummy taxonomies from the applicable list of dummy taxonomies which we obtained in the last part. Then, constructing a dummy data structure with two key-value pairs for each chosen dummy taxonomies where the first key, 'taxonomy', denotes one of the chosen dummy taxonomies, and second key, 'count', represents the number of occurrences of the corresponding taxonomy.
In Figure 3.5, the probability of adversaries getting the top one taxonomies, Sports, is 100% and is reduced to 50% in 3.6 because of two different taxonomies, Sports and Food, occurring the same times. Same situation happens to the rest taxonomies. So the methodology guarantees a 50% improvement in obfuscating top taxonomies.

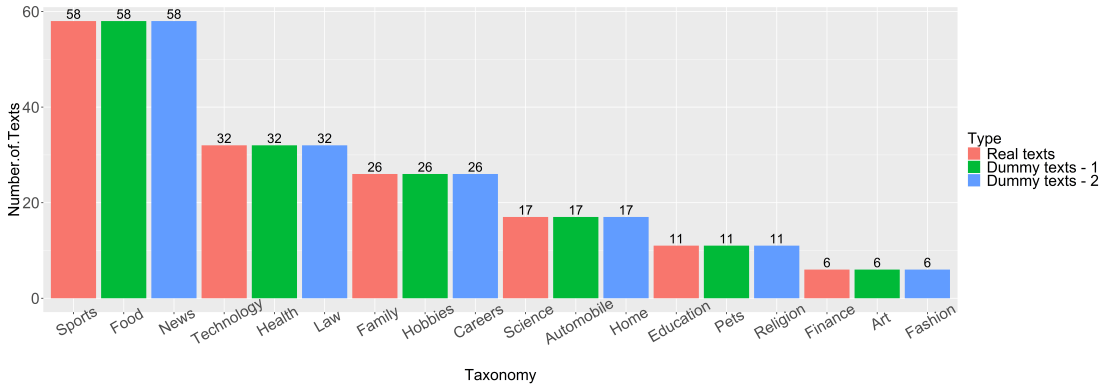


Figure 3.7 – Demonstration of post texts distribution while adding more dummy taxonomies

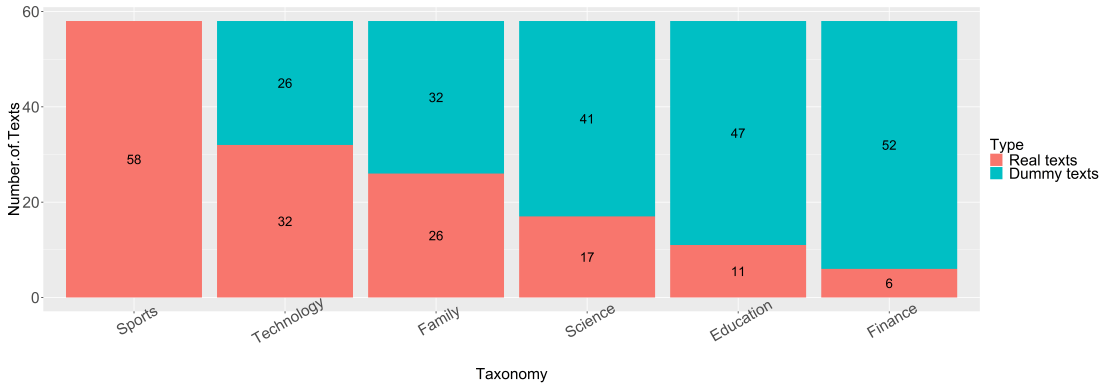


Figure 3.8 – Demonstration of post texts distribution while making occurrences of all taxonomies identical

To increase the difficulties for adversaries to know the real taxonomies, one could add more than one dummy taxonomies for a real taxonomy and result in the histogram shown in Figure 3.7 or make all taxonomies have the same appearances, as demonstrated in Figure 3.8. In Figure 3.7, the probability of getting the real taxonomies is decreased from 100% to 33%. And in Figure 3.8, the improvement of obfuscating top one taxonomies is increased to 83.3%. Although these two strategies achieve better information obfuscation, the execution time is much higher than the original strategy.

3.2.5 Generating dummy texts

After all the preprocessing, we could start to generate dummy texts. The Markov chain model text generator is operated in the same way as mentioned in Example 5. In our method, we adopt bigram Markov chain model instead of trigram, quadrigram or higher ones. The reason is that the trigram and higher gram would result in the dummy sentences generating from the same news reports or same sentence, which is what we try to avoid. The dummy sentences are supposed to be as random as possible without losing the likelihood of human-written

sentences and the situation that dummy sentences come from the same place would violate the rule of randomization. The pseudo code of this part is presented in the following,

for(each element in the dummy taxonomy data structure)

- The number, element.count, of dummy texts with a specified taxonomy, element.taxonomy, is generated via invoking Markov chain model text generator.
- At least two hundred words are generated in a text.
- The beginning letter must be capitalized and the ending letter is either a dot, a question mark, or an exclamation mark.

These dummy texts are distributed in several dummy folders. The number of dummy folders is equal to the square root of the number of dummy texts. The purpose of scattering the dummy texts is to prevent putting all eggs in one basket. That is to say, the probability of having the folder dummy is not $1/\#(folders)$, but is $\#(dummyfolders)/\#(folders)$. Moreover, keeping dummy texts in dummy folders could avoid mixing them with real texts and folders since G-Users might be confused if there are some texts which they are not familiar with existing in Google Drive.

3.2.6 Sharing dummy texts

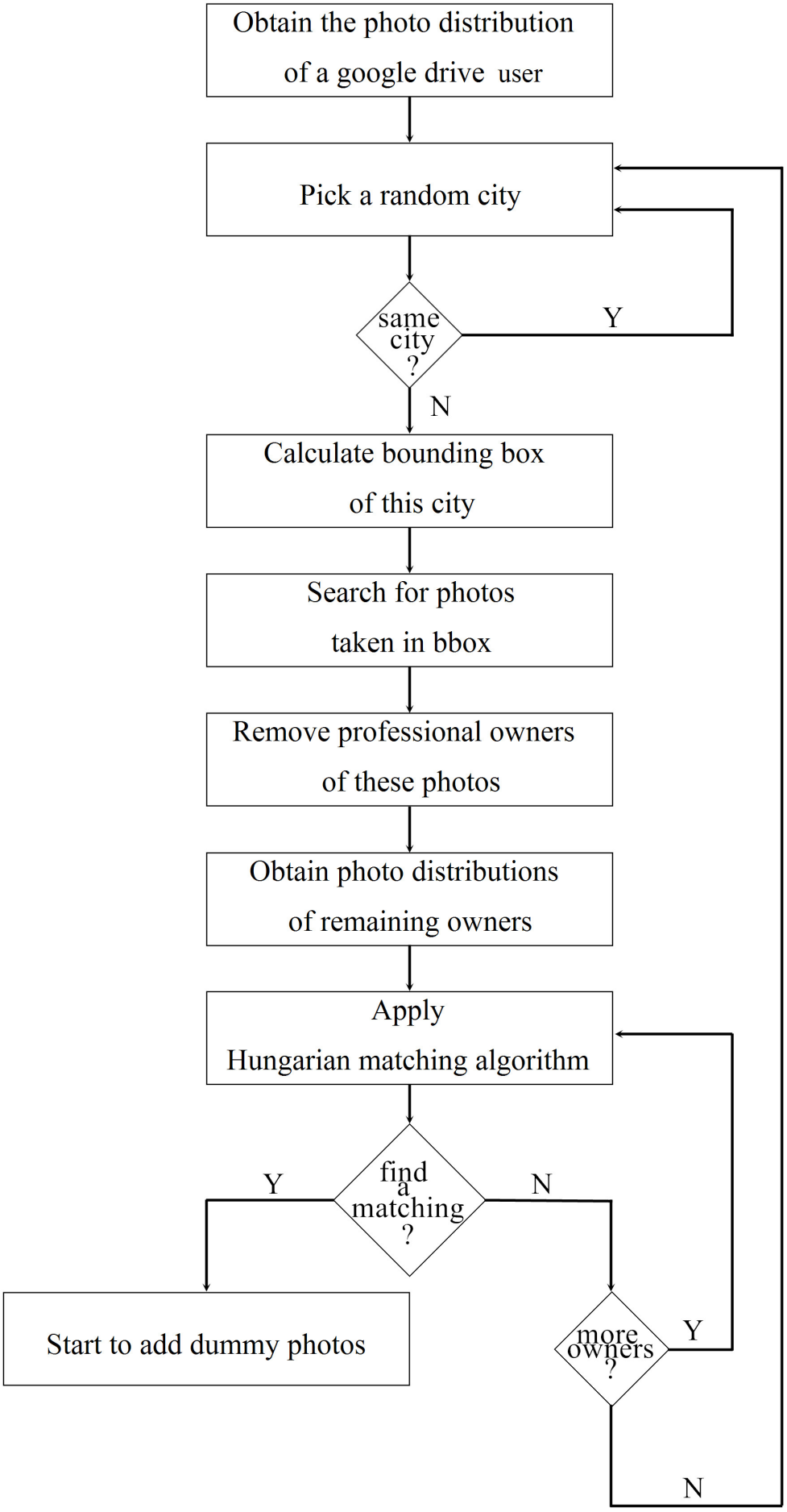
The last step is to share the dummy texts with one of the Google Drive users in order to make the dummy texts persuading. If we do not share dummy texts with others, all the shared texts will be regarded as real files. Besides, sharing dummy texts or folders is an obfuscation to Shared Interest. Hence, sharing dummy texts is necessary.
The way to share dummy texts is to randomly pick a registered users in the website [20] and share a dummy folder with that user. If sharing with nonmembers of our services, the dummy files might be shared with adversaries, which would easily compromise the effect of our methodology.

4 Methodology for Dummy Images

In Chapter 3, we talked about the method to generate dummy texts. One important feature in previous chapter is the construction of dummy taxonomy distributions. In this chapter, the proposed methodology for dummy images is presented. And the dummy image distributions also play important role in the process of adding dummy images.
The idiom "A picture is worth a thousand words" explicitly expresses the delicacy of proposed algorithm in adding dummy photos. The idea behind the proposed algorithm is to find a doppelganger [24] or a kagemusha [26] of the G-User. A doppelganger or a kagemusha refers to the double of another person. However, we are not going to find the guy who has a similar outer appearance of the G-User but the similar photos distribution of the G-User.
The information of images we intent to obfuscate is Face on Map, which is mainly determined by top locations and top faces. Hence, in this chapter the dummy images are added in the perspective of top locations and top faces. Since the top faces make sense only in the presence of top locations, we focus on the images which is either tagged with geography location or tagged with geography location and face.
The whole flowchart of the algorithm is shown in Figure 4.1 where each block is explained in more details in the following sections.

4.1 Real photo distributions of a Google Drive user

Before getting the photo distributions, we have to analyze the images of a G-User. The analyzer is implemented by taking advantage of AlchemyAPI and the results are stored in our database. The analysis is a preprocessing step for each member in our website.
In order to obfuscate the insights in photos, we have to get photo distributions of a G-User as the work we did in subsection 3.2.2. We try to obtain the histogram similar to 3.5. The initial task is to query the G-User in our database. The query to the database is shown as follows:


Obtain the photo distribution of a google drive user
Pick a random city
same city ?
Y
N
Calculate bounding box of this city
Search for photos taken in bbox
Remove professional owners of these photos
Obtain photo distributions of remaining owners
Apply Hungarian matching algorithm
find a matching ?
Y
N
Start to add dummy photos
more owners ?
Y
N

```
db.find
({
      'fileKind': 'image',
      'imageAnalysisResults.locations' : {$gt: []},
})
```

However, there is a problem about granularity of locations in the returned results. That is, some locations are expressed in a city level and some are denoted with a street level. For example, Renens VD is located in Lausanne city and they are separated with five kilometers only. It would be not wise to label them differently in the implementation. Hence, we have to do a clustering over the locations of the returned results. The algorithm of clustering is presented as below:

```
if( array is empty )
    Add the location to the array.
else{
     if( distance(new location, any existing location) is less than or equal to 10 km )
         Label the photo of new location as the same cluster.
     else
        Add the location to the array.
}
```

The algorithm applies to all the returned results. The distance function mentioned in the algorithm refers to the distance between two sets of coordinates. And distance threshold, default value is set to ten kilometers, is an input parameter. Since we deal with photos which are tagged with faces and locations as well as tagged with locations only, after clustering, the returned results are organized as the following data structure:

```
[
  {'location': 'Lausanne',   'withFace': 26,   'withoutFace': 34},
  {'location': 'Geneva',     'withFace': 20,   'withoutFace': 28},
  {'location': 'Bern',       'withFace': 18,   'withoutFace': 22},
  {'location': 'Zurich',     'withFace': 12,   'withoutFace': 24},
  {'location': 'Basel',      'withFace': 6,    'withoutFace': 6}
]
```

The returned results could be visualized in a histogram shown in Figure 4.2. Third-patry apps could have this visualization as well. If no dummy photos are added, the probability of adversaries getting the real top locations together with faces or without faces is 100%. Therefore, our goal is to add dummy photos to obfuscate the visualization. The source of dummy photos is flickr [8], which is a popular multimedia hosting website.

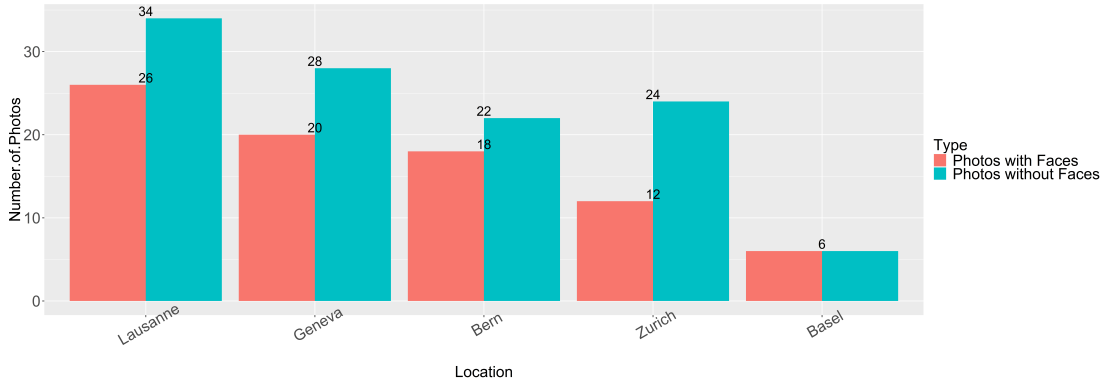


Figure 4.2 – Photo distributions of a G-User.

4.2 Selection of cities

Since we try to find a doppelganger of the G-User and do not know where the doppelgangers might live, we start by choosing a random city which, however, must not be identical to any one of the top locations we have discussed in section 4.1, otherwise the top locations still remain the same after adding the dummy photos.

4.3 Calculation of the bounding box

Now, we have an applicable city where the doppelgangers may live. Then search for the bounding box of this city with a radius equal to forty kilometers. The idea is that we look for doppelgangers living in this bounding box.
A bounding box, demonstrated in Figure 4.3, is specified by two sets of coordinates, one is at the upper right corner and the other is at the bottom left corner.

4.4 Photos taken inside the bounding box

With the bounding box, we can search for photos which were taken inside the bounding box. Since we do not know the doopelgangers' flickr id, we therefore query the owners of photos which were taken inside the bounding box.
The query we use for flickr api is:

```
flickr.photos.search(
      bbox: {22.7456, 120.34466, 23.2543, 120.8973},
)
```

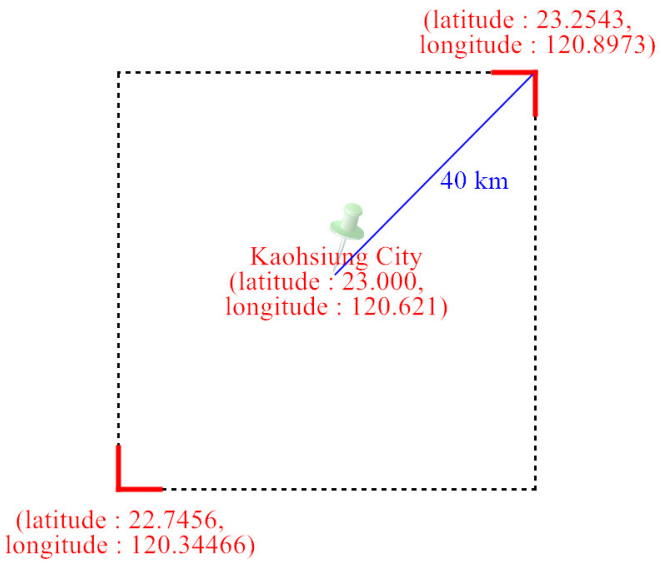


Figure 4.3 – The bounding box of Kaohsiung city

4.5 Filtering of owners of photos

Removing professional photographers is an optional function here. The algorithm to remove the professional photographers is stated in the following:

```
for(   each owner obtained from the above block    )
      flickr.people.getInfo(
            userid: flicrId
      )
       if(   the isPro field in the returned query is 1    )
            Remove the owner
      )
)
```

Since most works of photographers are sceneries and repetitive faces are rare, it is not natural that the G-User would have the same photo distributions as professional photographers. However, from time to time professional photographers contribute to the candidate pools of doppelgangers.

4.6 Photo distributions of a flickr owner

Until now, we have a candidate pools of doppelgangers who live inside the bounding box. Then we start to construct the photo distributions for each candidate. The process is similar

to 4.1. Firstly, gather the locations of photos by querying with flickr api. Since we are dealing with photos with faces and without faces, there are two queries. One is to query photos with faces and the other is without faces. And the queries we use are stated in the following block where first query is for photos with face and the second query is for photos without faces.

```
flickr.photos.search(
      {user_id: flickrId,
      has_geo: '1',
      tags: 'people, portrait, face' }
)

flickr.photos.search(
      {user_id: flickrId,
      has_geo: '1' }
)
```

After gathering the information of photos owned by one candidate, we perform the clustering algorithm to solve the granularity of locations. A data structure like the following is then constructed.

```
[
  {
    'location': 'Kaohsiung',
    'latitude': '23.000',
    'longitude': '120.621',
    'photoId': {'withFace': [12, 66, 52,63, ...], 'withoutFace': [43, 68, 17, ...]}
  },
  {
    'location': Taipei,
    'latitude': '25.086',
    'longitude': '121.560',
    'photoId': {'withFace': [11, 55, 89, ...], 'withoutFace': [19, 65, 88, 93, 96, ...]}
  },
  {
    'location': Tainan,
    'latitude': '22.981',
    'longitude': '120.94',
    'photoId': {'withFace': [10, 91, ...], 'withoutFace': [77, 76, 60, 62, 51, 53, ...]}
  }
]
```

Then we can visualize the data structure in a histogram shown in Figure 4.4.

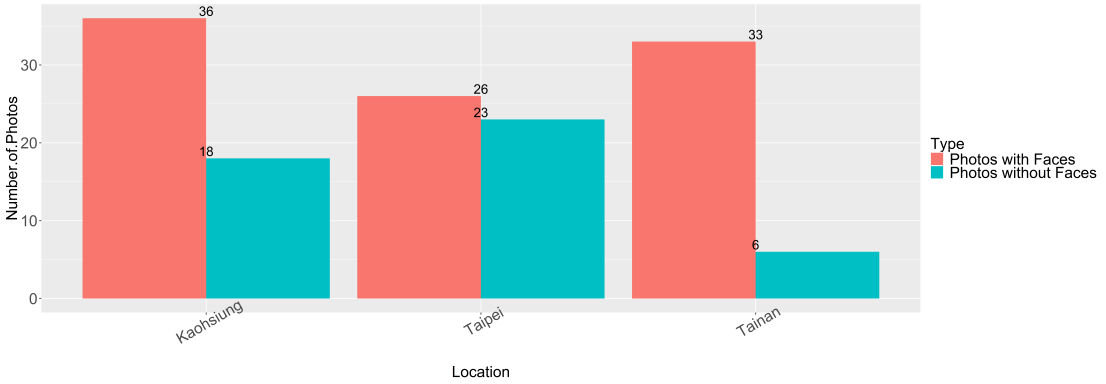


Figure 4.4 – Photo distributions of a candidate

4.7 Hungarian matching

We now have photo distributions of a G-User and candidates of doppelgangers respectively. Then how do we decide who the doppenganger is? Actually, we can solve the problem by applying Hungarian matching algorithm [25], also know as bipartite matching algorithm. The typical matching problem is online dating. Suppose in an online dating website where only ladies could choose the favorite gentlemen. And the goal of the website administrators is to maximize the matching pairs. The problem could be visualized in Figure 4.5 where red dash lines refer to the preferences of ladies over gentlemen and blue lines represent the successful matching pairs. The matching problem could also be found in other applications such as network flows, scheduling, work assignments and so on.
The matching algorithm could also be used to solve our problem. Figure 4.6 is the visualization of the simplified problem in which red dash lines refer to plausible matching solutions while blue lines denote the final solution. For example, photos located in Zurich has three applicable matchings which are photos situated in Kaohsiung, Chiayi and Hualien. And the final matching of Zurich is Hualine. The red dash lines are determined by the following algorithm and the blue lines are completed by Hungarian matching algorithm.

```
for( LCity in top locations of the G-User ){
    for( RCity in all locations of a candidate ){
        if( (#{photos of withFace in LCity}      ≤ #{photos of withFace in RCity}) and
            (#{photos of withoutFace in LCity} ≤ #{photos of withoutFace in RCity}) ) {
            Connect LCity with RCity with a red dash line.
        }
    }
}
```

In the above algorithm, the reason that the number of photos with faces in LCity should be less than or equal to the number of photos with faces is that we should add as many dummy photos as the original photos. As demonstrated in Figure 4.6, the photos situated in Lausanne can

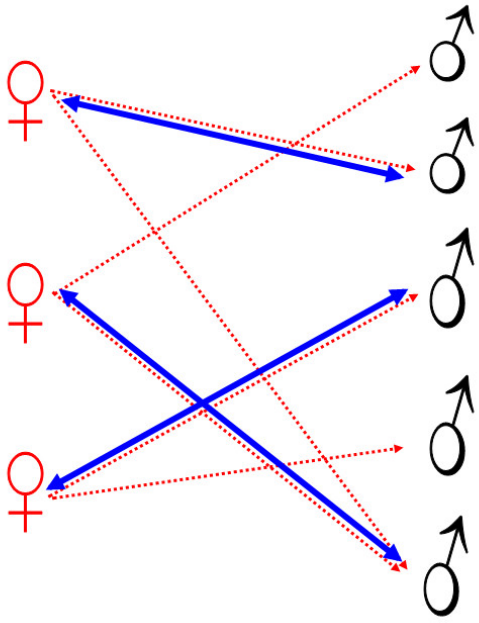


Figure 4.5 – Demonstration of matching problem: dating

only be matched with photos located in Kaohsiung. The photos of cities other than Kaohsiung fail to generate twenty six photos with faces and thirty four photos without faces. Hence, there is a red dash line only when the criteria is met. We call this method, strict matching. There is another method which we call loose matching in which a red dash line is presented when the following criteria is met, (#{photos of withFace in LCity} ≤ #{photos of withFace in RCity} - 2) and (#{photos of withoutFace in LCity} ≤ #{photos of withoutFace in RCity} - 2). The strict matching guarantees the number of dummy photos is the same the number of real photos. However, the loose matching is more time-saving in finding a matching. While implementing Hungarian matching algorithm, we adopt the loose matching.
Since Hungarian matching algorithm deals with cost matrix, in implementation the red dash lines are replaced with small positive value, for example 1, while the rests are filled with a big value, for example 100. If a candidate does not have a matching with the G-User, repeat the algorithm with another candidate until a matching is found. Sometimes all of the candidates do match with the G-User in the chosen city, then go back to 4.2 and follow the flowchart in Figure 4.1 again.

4.8 Uploading dummy photos to the Google Drive user

Up to this section, the doppelganger is found for the G-User and dummy photos could be uploaded into the Google Drive of target G-User. As we present in 4.7, the strict matching guarantees the number of photos owned by the doppelganger is larger than or equal to the number of photos possessed by the G-User. The dummy photos are selected randomly out of

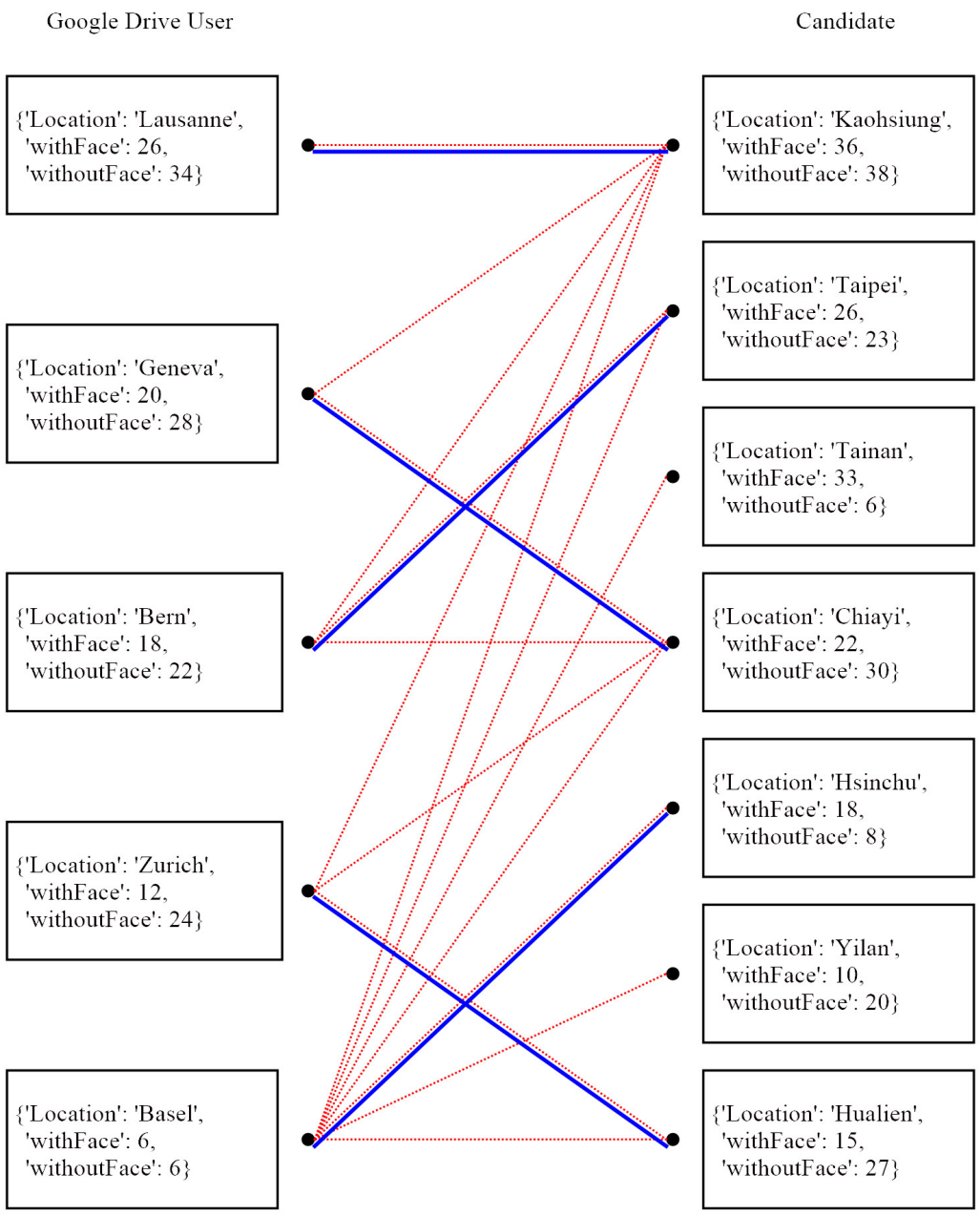


Figure 4.6 – Visualization of our matching problem

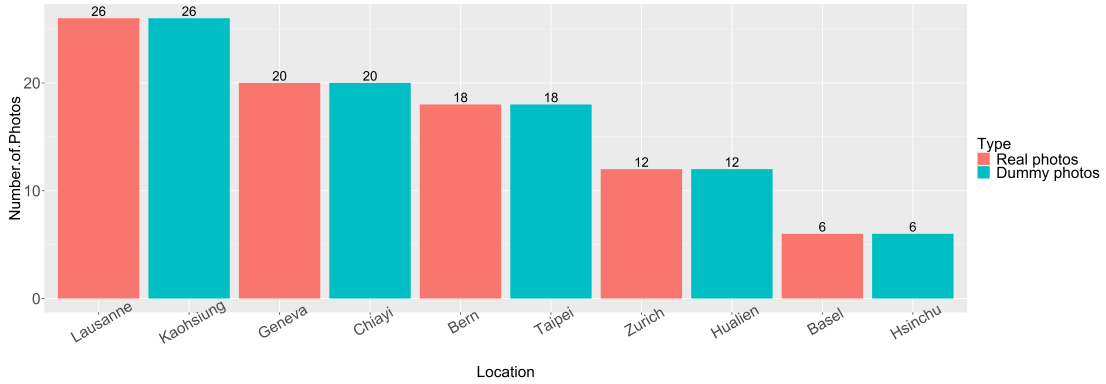


Figure 4.7 – Photo distributions after adding dummy photos with faces

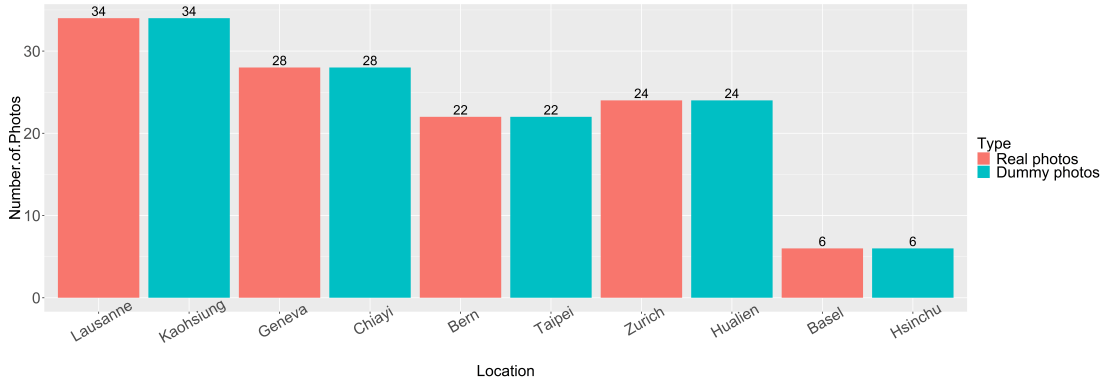


Figure 4.8 – Photo distributions after adding dummy photos without faces

the corresponding matching pair. While in the loose matching, the dummy photos are selected as many as possible if there are fewer than required. Let's take Figure 4.6 as an example, the dummy photos for city Geneva are chosen from the city Chiayi, where twenty out of twenty two photos with faces and twenty eight out of thirty without faces.

The histograms right after adding the dummy photos are supposed to be like the one in Figure 4.7 and Figure 4.8. And that is the insights third-party apps would get afterwards. In fact, we could find more than one doppelgangers for the G-User and add more dummy photos. For example, in Figure 4.9 and Figure 4.10, there are two doppelgangers instead of one.
The dummy photos are kept in a dummy folder such that the G-User would not be confused by the dummy photos. There is one more thing that is required to be noticed. The location may not be specified in metadata while downloading the photos. Flickr has a strict constraint and privacy protection over photos, which allows users to set the license level and sharing of metadata. Although cameras, smart photos and tablets could insert the coordinates of user's current location in Exif metadata, most users turn off the automatic positioning function. Fortunately, most users would indicate the location where the photo was taken while uploading it to flickr. Therefore, before adding dummy photos to google drive of the G-User, we have to insert the API-provided metadata in Exif metadata manually once the coordinates were not

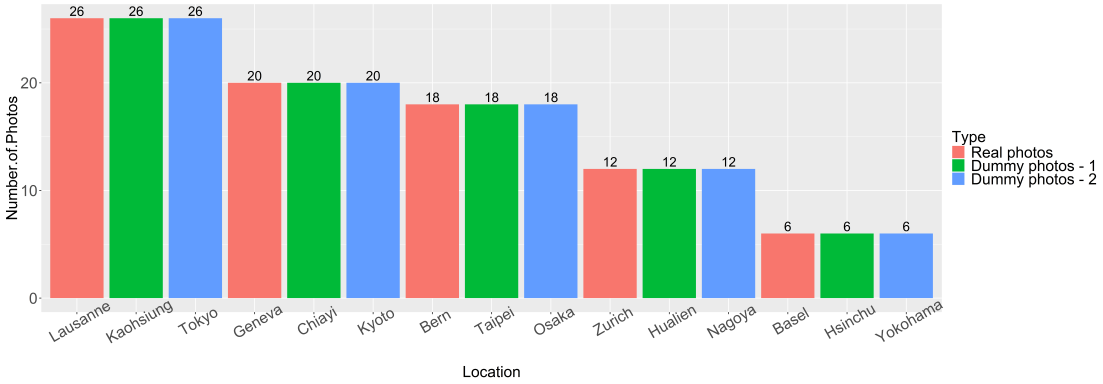


Figure 4.9 – Photo distributions after adding dummy photos with faces from two doppelgangers

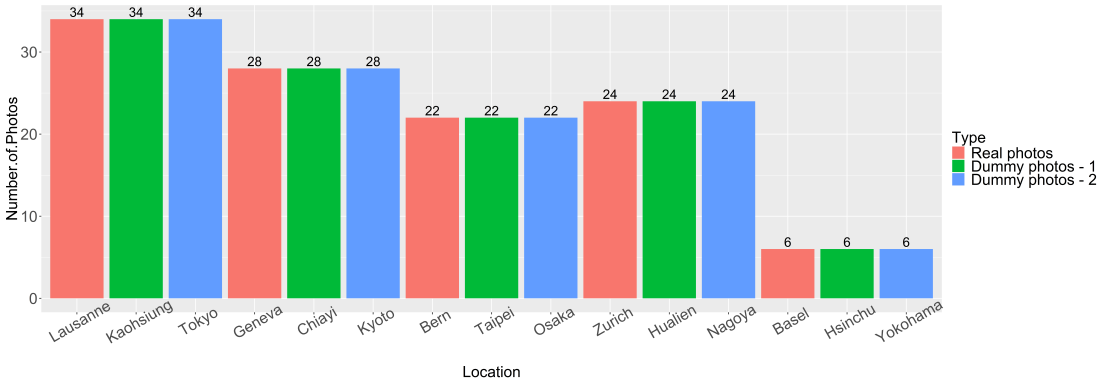


Figure 4.10 – Photo distributions after adding dummy photos without faces from two doppelgangers

written in the Exif metadata field at the beginning.

4.8.1 Sharing dummy photos

After dummy photos are done uploading to Google Drive of the G-User, we share the dummy photos with a random Google Drive user as the work in 3.2.6. Then, the process of adding dummy photos is finished for one Google Drive user.

5 Experimental Results

In this chapter, we present the experimental results in which three defined performance metrics are applied to measure the performance.

5.1 Experimental setup

The experimental environment is based on Node.js[17] and MongoDB[16]. Besides, there are some modules and api library are required. The following subsections will cover these tools.

5.1.1 Node.js

Node.js, developed by Ryan Dahl, is a back-end platform implemented in Javascript. The advantages of Node.js are efficiency and lightweight because of an event-driven and non-blocking I/O model. Moreover, Javascript developers do not need to learn another back-end languages to build servers. In a general way, Javscript alone could deal with front-end and back-end developments. When it comes to installing and managing packages, npm[18] is the perfect tool.

5.1.2 MongoDB

Database is an indispensable part in a server. MongoDB[16] is a NoSQL structure and is compatible with Node.js. This is an easy choice of database while the server side is implemented in Node.js.

5.1.3 Some important packages

In addition to Node.js and MongoDB, there are some packages which are needed to be installed. The most important among them are async, google api, and flickr api.
Although Node.js is launched in an asynchronous way, we sometimes demand the code to be

executed synchronously. And async[11] guarantees a serial execution.
Since we need to add dummy files in google drive for users, google drive api [14] is required.
In order to query photos, flickr api[7] is definitely needed as well.
Before using google drive api and flickr api, we have to register firstly and then obtain the unique id, key and secret.

5.2 Performance metric

To measure the performance of our method, we define three performance metrics. One is to indicate the percentage of dummy files third-party apps could get in the top taxonomies, locations or faces and another is to mention the fraction of dummy taxonomies, locations or faces in the top five and the other is to measure the shift of ranking of top faces and top locations.
We could formulate the first metric, dummy ratio (DR), with the following formula,

$$DR = \frac{\#(dummy \quad files)}{\#(total \quad files)} \tag{5.1}$$

This performance metric is to measure the probability of fetching a dummy file while third-party apps start to profile the user. We could use DR on the example files in Chapter 3 and get the results in Table 5.1.

Table 5.1 – Performance Matrix : DR

Files	**DR**
Texts in Figure 3.6	$\frac{58+32+26+17+11+6}{2*(58+32+26+17+11+6)} = 50\%$
Photos with faces in Figure 4.7	$\frac{26+20+18+2+6}{2*(26+20+18+2+6)} = 50\%$
Photos without faces in Figure 4.8	$\frac{34+28+22+24+6}{2*(34+28+22+24+6)} = 50\%$

In Figure 3.6, Figure 4.7 and Figure 4.8, it is obvious that DR = 50% since we added the same number of dummy files as the real files.
The second performance metric, dummy fraction (DF), is expressed as follows,

$$DF = \frac{\#(dummy \quad taxonomies)}{\#(top \quad taxonomies)} \tag{5.2}$$

Let's take Figure 3.6 as an example, DF is equal to 0.4. Since taxonomy 'Family' and taxonomy 'Hobbies' have the same number of texts. DF could be equal to 0.6 if taxonomy 'Hobbies' is ranked at five instead of six. Equation 5.2 is used to analyze texts rather than images. The reason is that one dummy text is likely to have more than one and real taxonomy while images are not.
The third performance metric, normalized ranking shift (NRS), is defined in the following equation,

$$NRS(f) = ranking_{post}(f) - 2 * ranking_{prior}(f) \tag{5.3}$$

where f represents one of the top taxonomies, locations, or faces. While $ranking_{prior}()$ refers to the ranking before dummy files are added and $ranking_{post}()$ denotes the ranking after dummy files are added. NRS tells us to what extent the top locations, or faces degrade in the ranking after dummy files are added. Let's take Figure 4.7 as an example. Table 5.2 demonstrates the prior ranking and pose ranking of each locations in Figure 4.7.

Table 5.2 – Prior and post ranking of locations in Figure 4.7

Locations	**Prior ranking**	**Post ranking**
Lausanne	1	1
Geneva	2	3
Bern	3	5
Zurich	4	7
Basel	5	9

By using Equation 5.3, NRS could be obtained in Table 5.3.

Table 5.3 – NRS of top locations in Figure 4.7

Locations	**NRS**
Lausanne	1 - 2 * 1 = -1
Geneva	3 - 2 * 2 = -1
Bern	5 - 2 * 3 = -1
Zurich	7 - 2 * 4 = -1
Basel	9 - 2 * 5 = -1

The bigger NRS is, the better obfuscation we get. If one top item has NRS which is lower than -1, then the top topic does not change its ranking. Therefore, NRS is a brilliant indication of shift of ranking for every top topic. In addition to NRS of individual location or face, we could

calculate the average NRS (avgNRS) which is expressed in Equation 5.4

$$avgNRS = \frac{\sum_{f=1}^{F} NRS(f)}{F} \tag{5.4}$$

In Equation 5.4, F denotes the number of top locations or faces we concern. From Table 5.3, the avgNRS is therefore -1.

5.3 Results

Mr. Hamza Harkous constructed a website [20] for users to realize the condition of possible leakages of their privacies. Every user could access the service by validating the permission requests which include accessing and managing their Google drive. And now we have over one thousand registered users. The following experiments will test over twenty users who have the most amounts of texts or photos. The top twenty users for adding dummy texts may not be the same as the users for adding dummy photos.

5.3.1 Text part

According to subsection 2.2.1.1, a text has three types of information, entities, concepts, and taxonomies. Therefore, we analyze the three different types of all the dummy texts. Figure 5.1 is the visualization of the performance metric, dummy ratio (DR). The situation that at least 50% of the texts in the top five entities are dummy texts are happened in fourteen out of twenty users. While Figure 5.2 is associated with the performance metric, dummy fraction (DF). And there are only four out of twenty users whose text files are added less than two dummy entities. Hence, the proposed methodology guarantees a good obfuscation of entities.
Figure 5.3 is involved with top five concepts by performance metric DR. The fact that more than half of the texts are dummy files are found in fourteen out of twenty users. Performance metric DF along with top five concepts is plotted in Figure 5.4, which has the identical outcomes as Figure 5.3. From these two figures, we know there is also a high obfuscation of concepts with the proposed methodology.
Figure 5.5 and Figure 5.6, are visualizations of performance metric DR on top five taxonomies and performance metric DF on top five taxonomies respectively. In these two figures, we find out that over 60% are dummy texts in the top five taxonomies, but the fraction of dummy taxonomies is either 0.2 or 0.4, which is lower than expectation. The situation implies some dummy taxonomies have overlapped with the original top five taxonomies, even we manage to exclude this situation in the process of generating dummy texts. The reason is that any text, either real or dummy one, could have more than one taxonomy. Hence, the dummy texts not only contribute to the number of dummy taxonomies but also to the number of original top five taxonomies.

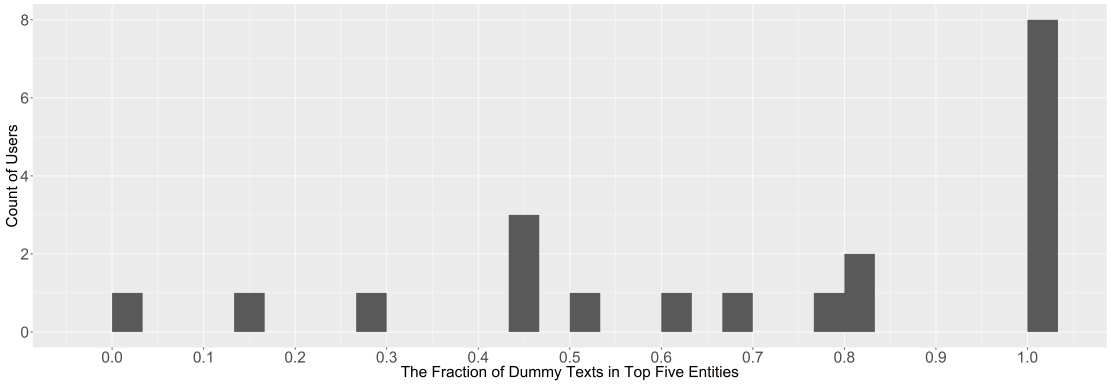


Figure 5.1 – Histogram of fraction of dummy texts in top five entities.

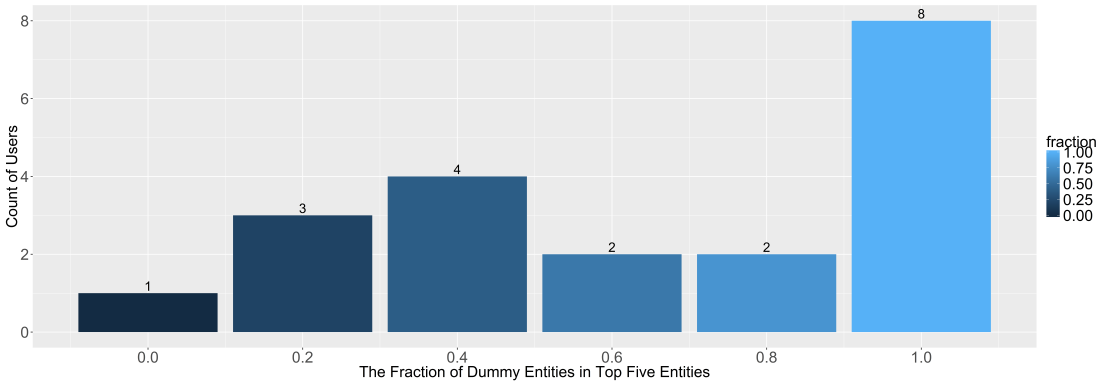


Figure 5.2 – Histogram of fraction of dummy entities in top five entities.

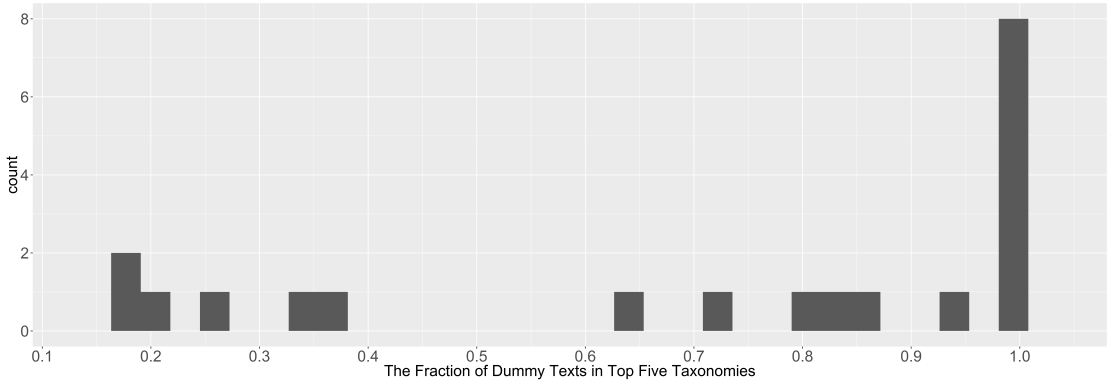


Figure 5.3 – Histogram of fraction of dummy texts in top five concepts.

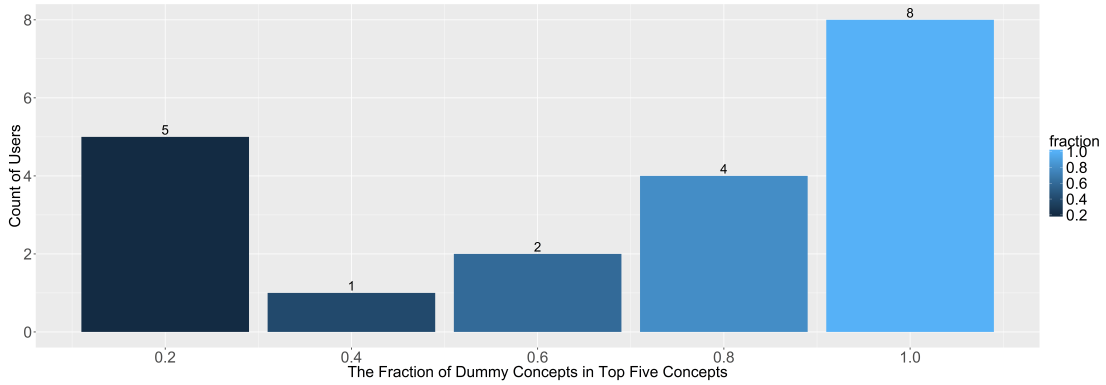


Figure 5.4 – Histogram of fraction of dummy concepts in top five concepts.

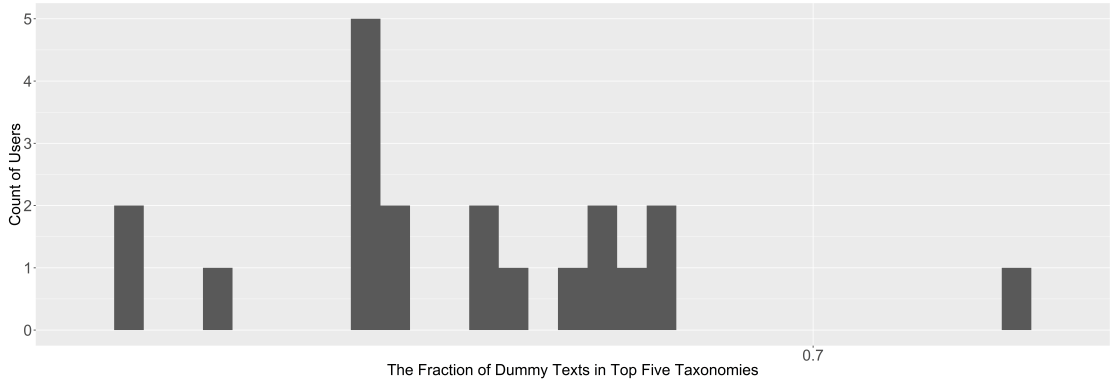


Figure 5.5 – Histogram of fraction of dummy texts in top five taxonomies.

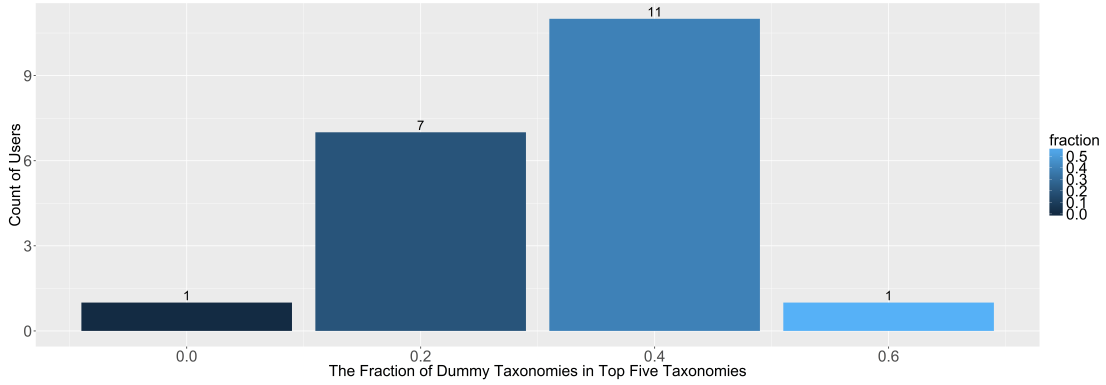


Figure 5.6 – Histogram of fraction of dummy taxonomies in top five taxonomies.

Figure 5.7 – Histogram of fraction of dummy photos in top five locations.

5.3.2 Image part

In the image part, we consider top five locations and top five faces if there are more than five locations or faces respectively; otherwise, take as many locations and faces as the user has. First, we present the outcome of top five locations. Figure 5.7 shows the performance metric DR on top five locations, Figure 5.8 refers to the performance metric DF on top five locations, Figure 5.9 denotes the number of NRS while Figure 5.10 is the histogram of average NRS. As we can see in these four figures, most NRSs are distributed between -1 and 1, which meets our expectation. And the fraction of dummy photos are more likely to be between 0.4 and 0.6 which is close to the theoretical value as well, 0.5 and most of the fraction of dummy locations are between 0.4 and 0.6. However, in Figure 5.7, there are two users who have zero dummy photo in the top five locations, which correspond to two outliers, 4 and 5, in Figure 5.10. The reason is that these two users whose photos were all taken in one location. Moreover, in Figure 5.9, the explanation for the situation that NRS occurs once at -2 and twice at -3 is the adoption of loose Hungarian matching. In the loose Hungarian matching, photos of dummy locations are added fewer than the photos of real locations, which might result in the lower ranking of a dummy location if the number of photos of real locations are very close. Overall speaking, the methodology matches the theory as well as successfully obfuscates the top locations.

Then we discuss the top faces. Figure 5.11 presents the distribution of fraction of dummy photos in the top five faces, Figure 5.12 shows the histogram of fraction of dummy faces in the top five faces, Figure 5.13 denotes the histogram of NRS, and Figure 5.14 depicts the distribution of fraction of average NRS in the top five faces. As we could observe in the four figures, NRS is distributed rather dispersedly and there are more outliers while comparing with figures of top locations. The reason is that the dummy photos are generated according to the constructed photo distributions which are location-oriented. Although, google drive users have a thorough face distributions in our database, flickr users are not likely to tag faces on the photos and flickr does not provide api related to faces. Hence, so far the methodology does not guarantee the obfuscation of top faces are as effective as that of top locations. Despite of

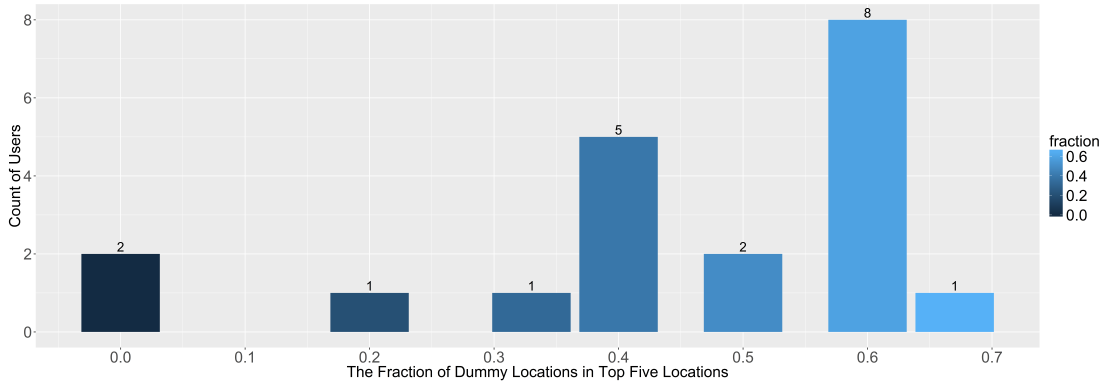


Figure 5.8 – Histogram of fraction of dummy locations in top five locations.

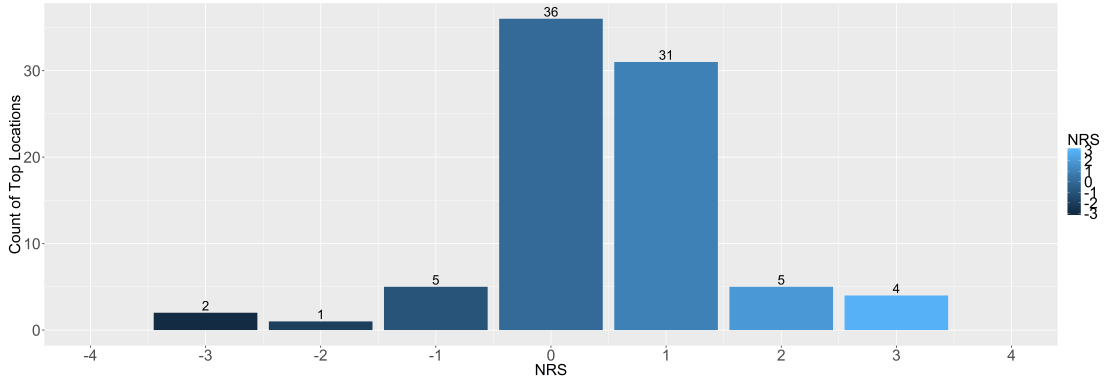


Figure 5.9 – Histogram of NRS in top five locations.

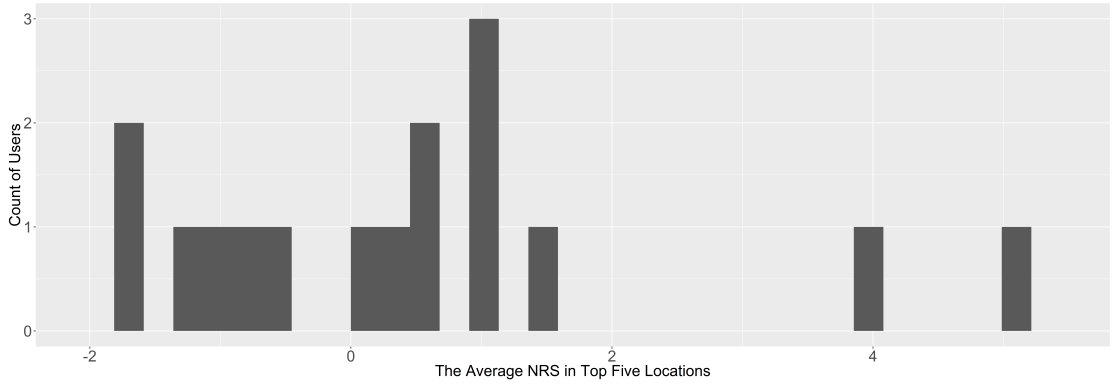


Figure 5.10 – Histogram of average NRS in top five locations.

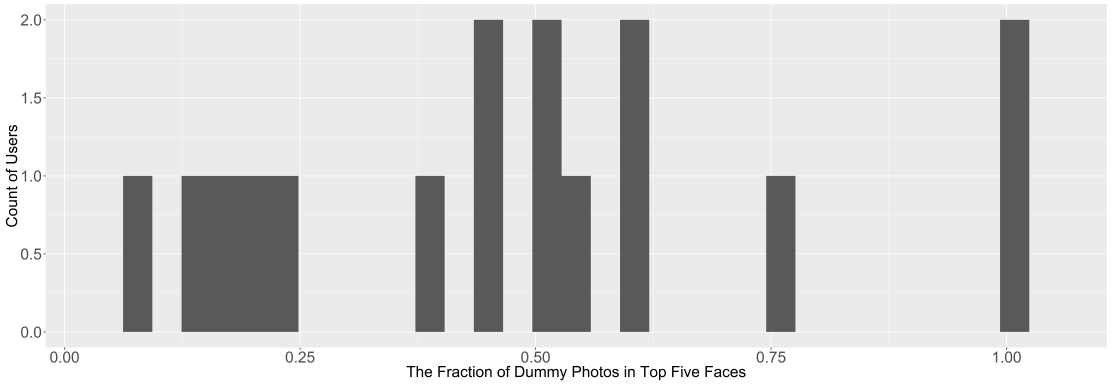


Figure 5.11 – Histogram of fraction of dummy photos in top five faces.

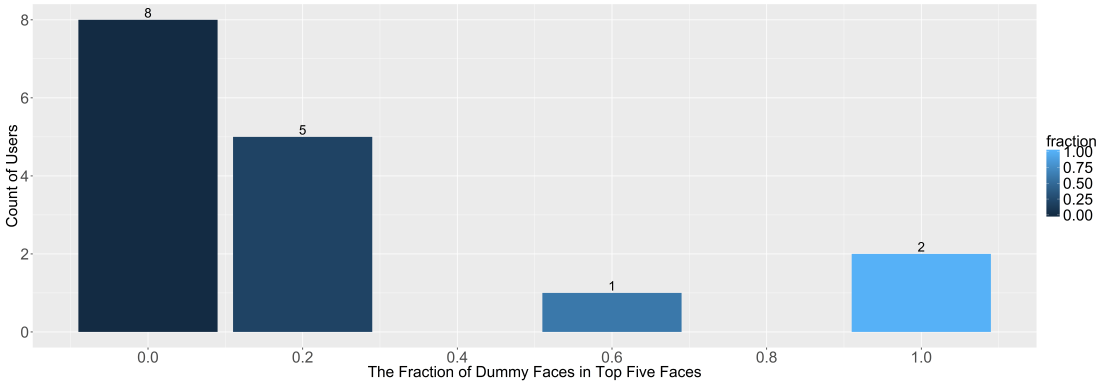


Figure 5.12 – Histogram of fraction of dummy faces in top five faces.

the disadvantage the proposed methodology might have, there are more than half of the users whose top faces are still obfuscated with dummy faces. Therefore, the proposed methodology proves to be an excellent method for obfuscating top locations and a little bit weak approach in obfuscation of top faces when comparing with the effect of obfuscation of top locations.

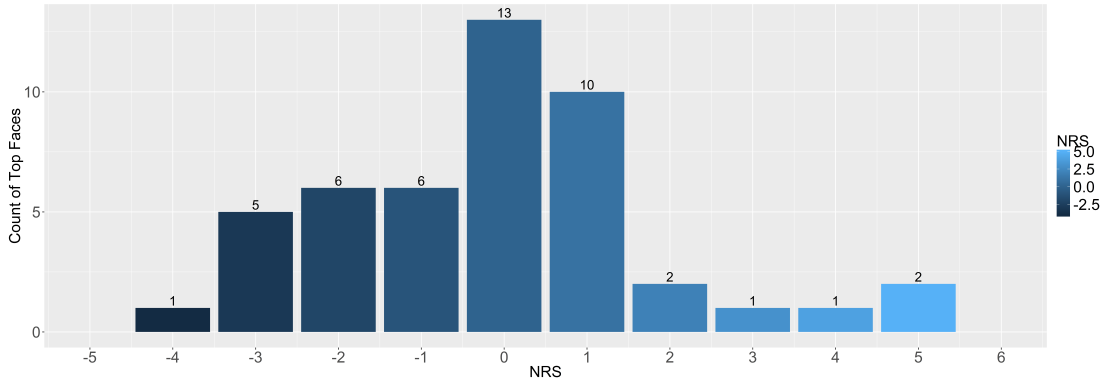


Figure 5.13 – Histogram of NRS in top five faces.

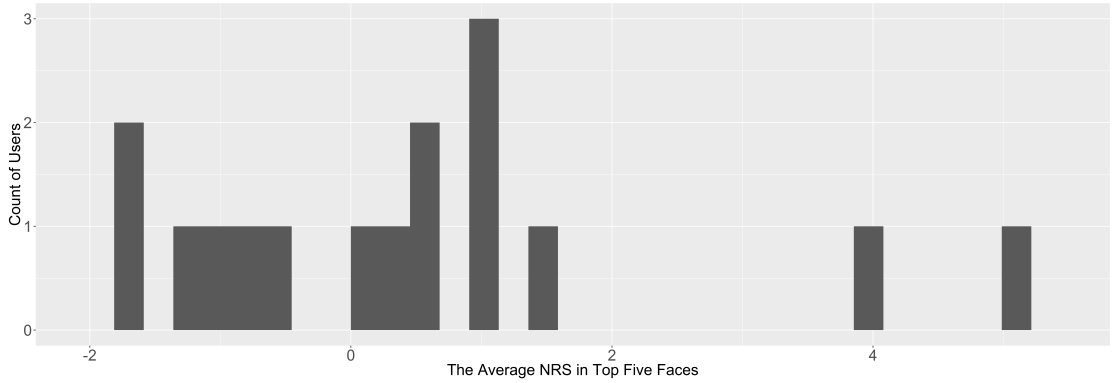


Figure 5.14 – Histogram of average NRS in top five faces.

6 Conclusions and Future Works

6.1 Conclusions

The realization of Moore's law, the powerful computing powers of portable electronic devices, the growing bandwidth of Internet and the increasing demands for convenient applications contribute to everyone's daily life in the digital era. As the technology advances in an unprecedented speed, people enjoy the convenience as well as worry about the leakage of personal privacy. Among all kinds of possible means of leaking personal privacy, third-party apps are the most dangerous entities that people should be aware of not only because of the easiness of installing and usage but also the binding of online personal accounts and these apps.
While people appreciate the amazing services third-party apps provide, their privacies are more or less open to the apps. Therefore, leakages of privacy are frequently and secretly taken place via the way that third-party apps analyze users' information to get insights of the victims. In this thesis, an effective methodology is proposed to obfuscate information by adding dummy texts and photos such that third-party apps would have troubles profiling their consumers. In the work of dummy texts, by taking advantage of bigram Markov chain model we generate dummy texts with taxonomies which are not covered in the lists of taxonomies in the user's original texts. While adding dummy photos, we construct a photo distributions of the user and search for the doppelganger in flickr website who has the identical photo distributions as the user. Afterwards, these dummy files would be shared with certain users for the purpose of diversity and more obfuscations of insights about top collaborators and shared interests.
From the experimental results presenting in Chapter 4, the methodology successfully served to obfuscate users' information according to three performance metrics, dummy ratio (DR), dummy fraction (DF) and normalized ranking shift (NRS), which are measurements of percentage of dummy files in the whole documents, fraction of dummy types and shift of ranking. The methodology shows a great obfuscation for top entities and concepts while a not bad obfuscation for top taxonomies. Since dummy texts come in more than one dummy taxonomy which may overlap with the real top taxonomies and result in less effective obfuscation as those in entities and concepts. However, the approach still guarantees a highly effective obfuscation in

texts. When it comes to images, the experimental results demonstrate a wonderful fulfillment of obfuscation in top locations and top faces. Although most flickr users are reluctant to tag faces on the photos along with the challenges that flickr does not provide face-related api, which leads us to develop an location-oriented algorithm and results in few outliers in the outcomes of obfuscation in top faces, the methodology successfully assists users in stopping third-party apps from getting insights of photos.
In a conclusion, third-party apps never stop profiling users and could get their insights which, however, are largely obfuscated by the proposed method. Although users might want to maximize the effect of obfuscation by adding more dummy files. Considering the execution time and the storage space in google drive a user could have, at the present stage we stick to the current methodology. There are some future works that need to be solved and will be discussed in the next section.

6.2 Future work

Despite of the success the current methodology has, some improvements and extra functionality are required in the future. Firstly, the obfuscation of top taxonomies may be compromised by the fact that a dummy text has more than more dummy taxonomy which is likely to contribute to the real top taxonomies. To avoid this problem, the source corpus must not be identified with one taxonomy but the all taxonomies it contains. Then the dummy texts are generated only via the source corpus which is perfectly matched by the dummy text distributions.
Secondly, to improve the the obfuscation of top faces to the same degree as top locations, the implementation of choosing the flickr photos which were taken during the same period of time might be helpful. Since we can not force flickr users to tage faces nor demand flickr api developing team to release a face api, the idea that repetitive faces are likely to be appeared in photos during a certain time period.
We would like to develop an extension for google drive file browser. With the extension, all the dummy files would be overshadowed or invisible to the users. This idea not only improve the experience of user usability but also maintains a clean and original file browser for users.
Finally, there is a future challenge that we could confront. That is the threat of peer-to-peer attack. We have to come up with some solutions to tackle this threat.

Bibliography

[1] Alchemy language api, url: http://www.alchemyapi.com/products/demo/alchemylanguage.

[2] Alchemyapi, url: http://www.alchemyapi.com/.

[3] Alchemylanguage features, url: http://www.alchemyapi.com/products/alchemylanguage.

[4] Alchemyvision - automatically extract and tag images, url: http://www.alchemyapi.com/products/alchemyvision.

[5] *Article about AdNauseam, url: http://cs.nyu.edu/trackmenot/TMN-Howe-Niss08-ch23.pdf.*

[6] A basic introduction to neural network, url: http://pages.cs.wisc.edu/ bolo/shipyard/neural/local.html.

[7] Flickr service api, url: https://www.flickr.com/services/api/flickr.photos.search.html.

[8] Flickr.com, url: https://www.flickr.com/.

[9] Github: Adnauseam, url: https://github.com/dhowe/adnauseam/.

[10] Github: An automatic paper generator, url: https://github.com/strib/scigen.

[11] Github: Async package, url: https://github.com/caolan/async.

[12] Github: Multi-layer recurrent neural network for text generator, url: https://github.com/karpathy/char-rnn.

[13] Google apps administrator help, url: https://support.google.com/a/answer/6105699?hl=en.

[14] Google developers for node.js, url: https://developers.google.com/drive/v2/web/quickstart/nodejs.

[15] Ibm watson developer cloud, url: https://www.ibm.com/smarterplanet/us/en/ibmwatson/developerclou

[16] Mongodb.org, url: https://www.mongodb.org/.

[17] Node.js, url: https://nodejs.org/en/.

[18] npm.com, url: https://www.npmjs.com/.

[19] Pdf reader app, url: https://play.google.com/store/apps/details?id=com.foobnix.pdf.reader.

[20] Privyseal, url: https://privyseal.epfl.ch/.

[21] Scigen - an automatic cs paper generator, url: https://pdos.csail.mit.edu/archive/scigen/.

[22] Wikipedia: Artificial neural networks, url: https://en.wikipedia.org/wiki/artificial_neural_network.

[23] Wikipedia: Contex-free grammar, url: https://en.wikipedia.org/wiki/context-free_grammar.

[24] Wikipedia: Doppelgänger, url: https://en.wikipedia.org/wiki/doppelg

[25] Wikipedia: Hungarian algorithm, url: https://en.wikipedia.org/wiki/hungarian_algorithm.

[26] Wikipedia: Kagemusha, url: https://en.wikipedia.org/wiki/kagemusha.

[27] Wikipedia: Markov chain model, url: https://en.wikipedia.org/wiki/markov_chain.

[28] Wikipedia: Recurrent neural networks, url: https://en.wikipedia.org/wiki/recurrent_neural_network.

[29] *Generating Sequences With Recurrent Neural Networks*, 2003.

[30] *CacheCloak: enabling realtime location privacy for mobile users*, volume 13, July 2009.

[31] *Hiding Stars with Fireworks: Location Privacy through Camouflage*, 2009.

[32] *Generating Text with Recurrent Neural Networks*, 2011.

[33] Helen Nissenbaum Daniel C. Howe. Official webpage of tracemenot, url: http://cs.nyu.edu/trackmenot/.

[34] Hamza Harkous. The curious case of the pdf converter that likes mozart: Dissecting and mitigating the privacy rist of personal could apps. Technical report, École polytechnique fédérale de Lausanne, 2015.

[35] Mushon Zer-Aviv Helen Nissenbaum, Daniel C. Howe. Official webpage of adnauseam, url: http://adnauseam.io/.

[36] Andrej Karpath. Github: The unreasonable effectiveness of recurrent neural networks, url: http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[37] Tomas Mikolov. Statistical language models based on neural networks. 2012.

[38] Sarah Mitroff. Onedrive, dropbox, google drive and box: Which cloud storage service is right for you? Novermber 9, 2015.

CHENG-HSIANG CHIU

chenghsiang.chiu@gmail.com
Allée du Tilleul 3, Chavannes-près-Renens, Switzerland
(+41) · 787 · 572 · 266

EDUCATION

École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland — Sep. 2013 - Present
Master of Computer Science

National Chiao Tung University, Hsinchu City, Taiwan — Sep. 2005 - Aug. 2007
Master of Science in Communication Engineering

National Chung Cheng University, Chiayi City, Taiwan — Sep. 2001 - Jun. 2005
Bachelor of Science in Electrical Engineering

RESEARCH EXPERIENCE

Protection against Data Profiling by Adversarial Cloud Applications — Sep.2015 - Present

Master Thesis Project
Design and develop protection mechanisms for protecting users data on personal cloud-storage providers against data analysis by 3rd Party Apps.

Bootstrap recommender systems with the crowdsourcing — Feb. 2014 - Jan. 2015
Semester Project

Proposed an explicit technique to alleviate the impact caused by cold start in recommender systems by taking advantages of crowdsourcing platform.

Random Walk — Feb. 2014 - Jun. 2014
Team Project

Used Python to implement a web crawler to gather public events and activities in Europe which are used as references for recommending travelling routes to tourists.

High Value Cargoes' Networks — Feb. 2014 - Jun. 2014
Team Project

Developed a framework for historians to add newly-found data and visualize the routes of high value cargoes during the period of the Venice Atlas

WORK EXPERIENCE

CERN, Genève, Switzerland — Mar. 2015 - Aug. 2015
Intern

Developed softwares to dicover devices on the network, perform consistency checking with the installation databases, and monitor the status of the data acquisition network.

Academia Sinica, Taipei, Taiwan — Sep. 2012 - Jun 2013
Research Assistant

Applied CUDA to accelerate the processing of bio images, especially brain images.

National Chiao Tung University, Taipei, Taiwan — Jan. 2009 - Jun 2013
Research Assistant

· Constructed a data center for storing brain images in a distant health care project.
· Designed an load balancing algorithm to streaming videos.
· Constructed wireless sensor networks to monitor structural health, such as bridges.
· Coordinated and built surveillance systems in a campus and gadgets to guide blind students.
· Constructed a multimedia streaming system and deployed it in a vision-based intelligent environment.

SKILLS

Languages	Mandarin (native), English (fluent)
Computer Languages	C/C++, Java, Python, Javascript, Node.js, PHP, HTML, CSS

PUBLICATION

Semester Project Report

· Cheng-Hsiang Chiu, "Bootstrapping recommender systems with the crowdsourcing II," École Polytechnique Fédérale de Lausanne, 2015.
· Cheng-Hsiang Chiu, "Bootstrapping recommender systems with the crowdsourcing," École Polytechnique Fédérale de Lausanne, 2014.

Master Thesis

· Cheng-Hsiang Chiu, "Process Control in Streaming Server," Master Thesis, Department of Communication Engineering, National Chiao Tung University, 2007.

Conference Papers

· Yi-Yuan Chen, Yuan-Yao Tu, Cheng-Hsiang Chiu, and Yong-Sheng Chen, "An Embedded System for Vehicle Surrounding Monitoring," *IEEE Conference on Power Electronics and Intelligent Transportation System*, 2009.
· Cheng-Hsiang Chiu, Pang-Chan Hung, Jen-Hui Chuang, and Shing-Lu Huang, "Object Tracking under Sensing Lighting Equipments," *IEEE Conference on Industrial Electronics and Applications*, 2010.
· Der-Cherng Liaw, Cheng-Hsiang Chiu, Chia-Wei Yeh, Chia-Ming Chang, and Hsiao-Jen Hsieh," A Server Load Balancing Design for Peer-To-Peer Network," *International Conference on Computers, Communications, Control and Automation*, 2011.
· Der-Cherng Liaw, Chia-Wei Yeh, Cheng-Hsiang Chiu, Chia-Ming Chang, and Hsiao-Jen Hsieh, "A Load Balancing Scheme for Web Server Design," *International Conference on System Science and Engineering*, 2011.
· Der-Cherng Liaw, Yi-Hung Hsieh, Jing-Hong Lai, and Cheng-Hsiang Chiu, "A Network Topology Design for Structural Health Monitoring," *Asian Control Conference*, 2011.
· Der-Cherng Liaw, Jing-Hong Lai, Cheng-Hsiang Chiu, and Jia-Hong Liao, "A Wireless Sensor Network Platform for Indoor Surveillance System," *International Conference on System Science and Engineering*, 2011.