SAPIENZA
UNIVERSITÀ DI ROMA

Evaluation of data inference methods on the Google+ social network

Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Computer Engineering

Candidate

Francesca Piccione

ID number 1191511

Thesis Advisor

Leonardo Querzoni

Academic Year 2013/2014

Ed eccomi qui, alla fine di un percorso di studi inziato circa 6 anni fa. Sono stati anni pieni di fatiche, di sacrifici, di giorni passati sui libri a studiare ma anche anni ricchi di soddisfazioni personali. Fortunatamente nei momenti difficili ho avuto sempre accanto persone che mi hanno spronato a non mollare mai, a portare avanti i miei obiettivi e a sostenermi quando le cose sembravano non andare bene. Dedico a queste persone speciali il lavoro svolto in questa tesi.

Grazie ai miei genitori, Pia e Filippo, per avermi sotenuta sempre sia economicamente sia moralmente. Devo a voi la maggior parte delle soddisfazioni ottenute durante questi anni. Grazie per esserci sempre stati.

Grazie alla mia dolce metà, Marco, per avermi fatto sentire la sua vicinanza e la sua presenza durante i momenti difficili, quando pensavo di mollare tutto. Semplicemente grazie per essere come sei...

Grazie a mio fratello Claudio, a mia cognata Erika e alle mie due splendide nipotine Chiara e Giulia. Grazie per esserci sempre stati in questi lunghi anni e per aver creduto in me.

Infine un ringraziamento particolare ai miei due amici pelosi che solamente con uno sguardo riescono a trasmettermi un' amore infinito.

Indice

Elenco delle figure

Capitolo 1

Introduction

Because of the increasing popularity of Web, and therefore with the development of Internet, many Social Networks as Facebook, Google +, to name a few, have emerged. In a first moment the principal purpose of these platform was to establish relationship among people, but today as user can create his own social profile where sharing some data and informations. These social platforms are becoming very popular among people, offering several advantages and services to the final users such as the possibility to stay connected with their own friends, mingle with others people having similar interests, share a lot of informations and chat online among the others. On these social platforms, many users are open to share personal informations, displaying personal attributes as geographic location, hobbies, interests and school attended, while other people utilize these social platforms to form friendship links and affiliation with groups of interest.

The possibility to share a lot of data and the increasing number of people that use the Social Networks, implies the need to manage the privacy of the

users. This topic is becoming an always increasing concern for these social platform.

Many Social Networks make available several options to set the privacy of a user, in particular the principal are the following:

- the data can be shared with all users of the social graph, therefore the attributes of a user are visible to everyone and there isn't a limit on visibility of the informations (*public attributes*);
- a user can decide to hide some of his own attributes, so these informations are visible only by the user that has declared them (*private attributes*);
- the informations can be shared only with a subset of users, often our friends, therefore the data have a limited visibility.

The possiblity to limit the visibility of the informations is very important for another question. There are many applications that collect many public data from the social profile. The principal purpose of this application is to make direct advertising to a specific consumer, therefore the public data has also an economic value.

The possiblity to decide whether to share or limit the visibility of some informations implies that not all the users provide these attributes on their own social profile. For this reason could there be the possibility that a malevolent user could decide to infer these private informations, basing on the probability that many users could decide to make their own data public instead of private. Based on this assumption there are important questions that need

an answer: it's possible to infer private attributes of a given user, using the public available data on a Social Network? In which way this could be possibile? Which may be the effects of this type of analysis?

The answers to these questions will be seen in the next chapters, but for the moment is important to understand the basic idea of this anaysis. As earlier said, the majority of the users decide to sign up on a Social Network to remain in contact with their own friends or to share a lot of data. In particular the possibility of a user to mingle with people that share the same interests, therefore establishing friendship link with similar people, represent the key to infer private attributes and therefore to realize this type of analysis. The idea is that attributes not declared by a user could be inferred using people that have a particular degree of similarity with him, taking advantage of the public available data. This concept is the point of departure of the project developed in this Master Thesis.

In the next chapters we shall discuss about the importance of the inference of private data, its possible effects and the results obtained until now. After this general overview will be described several methods to realize this type of analysis and in particular the basic algorithm used for these. Therefore will be introduced a description of a technique known as *Collaborative Filtering (CF)*, of which the basic idea is to find similar users elaborating public data to infer private data of a given user. The research of similar users not always can be made efficiently, in fact for this reason has been implemented an important and popular technique known as *Locality Sensitive Hashing (LSH)*.

Capitolo 2

Problem overview

In this chapter is described a general overview about the problem to infer private attributes on a *Social Networ*k, the importance of this problem for the society and the level of its development over the years, examining the *State of Art*.

2.1 Problem definition and motivations

The *Social Networks* are very popular among the people and one reason of this success is the possibility to share a lot of informations with other users of the same platform. Today the Social Networks are an important costant in the people's life that using them for several reasons, such as working issues or only for personal amusement. Althought the several adavantages that a Social Network could present, there is an important problem: the privacy of the users. Information privacy is one of the most urgent issue in the information systems because the data on the Social Networks are subjected to

high risk if they aren't managed in a good way and therefore the safety of the data not always could be guarantee.

Many people have the mistaken illusion that in the social platform their own privacy is guaranteed. This thought depends by the fact that several Social Networks, such as Google Plus and Facebook, to name a few, offer the possibility to set the privacy level of our social profile, deciding the visibility degree of the data. The principal problem is that many people ignore the possibility to change the level of privacy of their own social profile, therefore lot of data are public.

The public informations, that apparently a user not believes important and making visible to others users, represent a possible risk for his own privacy. In fact these public informations could be used in several ways to infer private data that a user does not shares or with limited visibility, violating his own privacy. This problem is very important because would mean to violate the rights of the people, dissemination of private data and knowing possible sensitive informations. Understanding the importance of the public informations available in the Social Networks is fundamental to prevent this problem and safeguard the privacy of the users.

An important purpose is that to sensitizing the final users about this problem and underlining the importance of the visibility of some informations. The Social Network should worry to sensitize the public opinion about this problem, make possible solutions about this type of analisys, but unfortunately the*y* don't show interest to this problem for obviuos reasons of convenience.

2.2 State of the art

Today the *Social Networks* are become always more present in the life of many people offering several services and opportunites. Whether on one hand these platforms offer several advantages, on other hand the possibility to share many informations is often a negative aspect for the privacy of the users. As said in the precedent section, these social platform offer the possibility to change the visibility of informations, so a user can decide the level of visibility on the base of his own needs. The majority of users underrate the importance to hide some data, allowing to several users to see these informations. In a first moment these attributes could appear without importance, but really they leak many useful *"informations"* to deduce attributes that a user wouldn't t to reveal on his own social profile. This type of analysis bases its force on availability of public data shared by users.

Over the years many algorithms have been proposed to infer private attributes through the public available data on a Social Network, with particular attention to discover which type of public informations could be more useful to infer private attributes, leaking useful informations for this purpose.

Some studies show the importance of the friendship lists [1][10], basing on the idea that the users establish friendship link with people that, with high probability, share their own interests. This concept is very important because the value of a private attribute could be research among the values of this type of people. In particular the values of the attributes, shared by these users on their own social profile, could be the same that a given user would have declared on his own social profile.

In particular has been implemented an algorithm, known as *PrivAware*, that measures the privacy risk on the Facebook Social Network using the friendship links of the users. This tool has been designed to execute within a user's profile to infer attributes of a user, provide reporting and quantify the privacy risk attributed to friend relationships. The principal purpose of this method is shown that the majority of sensitive attributes can be derived from social contacts and show the possible solutions to reduce the privacy risk associated with this threat. The basic idea of the algorithm is that for each private attribute, the algorithm easily selects the most popular value of this attribute among the user's friends. If the number of friends that declare this value is major than a threshold the algorithm assign the value to the considered attribute, otherwise the attribute isn't inferred. An important problem of this algorithm is the disambiguation of the possible values. There could be many values that refer to the same attribute, for example "Uniroma1" and "Sapienza" refer to the same University; which value should be assigned to the given attribute? The solution is to create a dictionary of the possible variations for some attributes, such as University, and the algorithm uses this dictionary to transform values into canonical forms. The results show that the friendship list is an important public information for the inference of private attributes, because the 50% of the considered attributes correctly are inferred. A possible solution for this problem is to hide the friendship list, some type of friends or adding fake friends. In particular deleting from the list of a user the friends with the most attributes and with the most common friends are valid solutions to prevent this type of analysis. Also adding fake friends could be a possible solution because these people are fake and

therefore with high probability the returned inferred value mismatches with the user's true attribute.

Another important ingredient is the possibility for a user to create affiliation with groups of interest [1] . As for the precedent case, the basic idea is the following: whether a user is present in a group means that the users of the group share similar preferences. This pecularity show the importance of the group that could contain relevant informations that could be elaborated to infer considerable data. For example, suppose that a user is present in a group cocern the city of Rome, but the attribute for this type of information is private. There are two possibilities: the user lives in Rome or he visited this city, but this only information is not sufficient to infer the value of this attribute. Suppose that the social friends of this user share this attribute on their own social profile. What means? Whether the majority of these users have declared Rome for the attribute city, with high probability the given user lives in Rome. This example should show as two informations public (friendship list and groups), apparently without importance, could be relevant for the inference of a private attribute.

The obtained results show that there is good inference using both friendship and groups and this is very important because very often these two informations are declared public on the social profiles.

Another important algorithm underlines the importance of the semantic correlation among the public data [2]. The principal idea of this method is always the same: the data apparently without importance, assume a foundamental role for the inference of private attributes if associated to a semantic knowledge. An important problem for this type of solution is that the public

data should be elaborated capturing their semantic correlation, but this task cannot be easily automated. The principal idea is to upgrade a given interest adding other informations, for example using *Wikipedia*[1], and poolling similar interests under the same set *(Latent Dirichlet Allocation, LDA)*[2]. In this way, the users that take an interest for the same set are similar and their own data are used to infer the private attribute of a given user.

Another possible approache introduce the concept of *community* about an attribute *[3]*. The users that share the same attribute, model a community around this element and through different metrics is important to evaluate the robustness of this community. This value is important to understand wheter it could be useful for the prediction of private attributes of some users. The idea is to evaluate the other members of a seed community that with high probability not have declared the considered attribute. The principal idea is that their private value is equal to the value around which the community has been build. The results obtained for this approach are very good as for the precedent cases.

All the precedent methods have common points that are very important, in particular the elements more significant are the following:

- the importance of the Social Network choosen for the inference of private attributes;
- the choice of the private attribute that should be inferred.

[1] is a online multilingual and free content encyclopedia.

[2] is a model that captures statistical properties of text document and puts together under the same set (Topic) the documents that present the same text properties.

These two items are related because for the inference of a private attribute are very important the quantity of informations available for it and consequently the Social Network considered. The informations available for an attribute could be more present in a Social Network than to another, because of the structure and the nature of the same. For example several results[1]show that some type of attributes are inferred better on a Social Network as *Facebook* than to *Flickr*. These differences depend by the different nature of the two Social Networks. *Flickr* is a social platform designed to share photos, while Facebook allow to share a lot of several informations such as our interests, hobbies and much more. For this reason the probability that some attributes have more values on a Social Network as Facebook is high.

Unlike the previous methods, thare are many algorithms that underline the importance of some application available on the Social Network to infer private sensitive data[8]. Data coming from social applications is a most available source of information that a malevolent user might draw attack to the privacy of individuals. *Google Latitude*[3] is an example of this application that allows to find in real time the current location of people through the mobile phone. This service was used on some Social Networks, such as Facebook, allowing to the users to localize the movements of their own friends who have previously agreed to this service.

Tha data should be sanitized to protect sensitive informations. In particular there are two methods of security:

- the first concerns the integrity of data. When a user modifies his own data, this change striclty should be controlled to gurantee the

[3]this service has been discontinued on August 9, 2013.

truthfulness of this data;

- the second is the protection of informations from inappropriate visibility. Phone number, address and name are examples of this type of data.

Some of sanitazation techniques consist to add details to the informations, useful to prevent that learning algorithms are enabled to infer private attributes of a user. Some type of details of a user should be deleted because could help the learning algorithms to predict personal details.
Another solution is to manage the link informations that can be manipulated in the same way of details. Is important to consider the effects of privacy removing the friendship links that leak many useful informations and could be elaborated in several ways to infer private attributes of the users.
The privacy of the users is intimidated also for another reason. In fact there is the problem of leakage of informations as a direct result of the actions of the same Social Networks [11]. A company, such as the elctronic arts, could require to a Social Network to obtain public data of the users to advertise some type of products such as possible games that interest the final consumers. Really this company want to use this informations to infer some private attributes of the users such as their own politic affiliation for lobbing effort. Therefore is important to explore how the online social network data could be used to infer private attributes that a user doesn't want to declare on his own social profile and the possible solutions to prevent this type of problem.
Some type of algorithms that use the *Naive Bayes* classifier[9][11] that assu-

mes that the presence (or absence) of a particular feature is unrelated to the presence (or absence) of any other feature. Therefore a *Naive Bayes* classifier considers all of the features independently to return a result. Some of the methods modify this classifier and use both node traits and link structure. Also in this method, as in the previuos algorithms, emerges the importance of the friendship list of a user. To protect the privacy of a iuser, in this algorithm has been implemented several tests in which have been deleted both some informations from a user's social profile and link details as the friendship link among users.The principal purpose of this algorithm is to study the effects that the knowledge of the precedent informations has for the inference analysis. In particular the results indicate that removing both trait details and friendship links together is the better way to prevent the inference of private data of a given user.

Through the *State of Art* have been possible understand which elements could be used in this Master Thesis. In particular the common element for the methods proposed in the precedent algorithms is the importance of friendship list. The obtained results show that the quality of the predictions are best using this public information and this result is expected because the probability that a user shares interest with his own friends is very high. The principal idea is that the possible value of a private attribute of a given user could be find among the values shared from his own friends and for this reason these people are very important. For this reason in this Master Thesis a method based on this idea has been implemented to demonstrate that the evaluated predictions are better than the predictions obtained for the other methods.

To evaluate the quality of the results, many algorithms used the Precision and the Recall metrics, that are used for the evaluation of the methods implemented in this project.

Thanks to the *State of Art* has been possible to understand which algorithms return good predictions, therefore which type of algorithms are more dangerous for the privacy of the users. This is very important to realize solutions to safeguard the privacy of the users on the Social Networks, which is always an important issue for the society.

Capitolo 3

Design and Implementation

In this Chapter are shown the principal elements used for the purpose of this Thesis. First of all will be described the method of *Collaborative Filtering (CF)* that represents the base of the implemented tecniques. Following, will be described the principal tecniques to evaluate the *Similarity* among users, the problem observed during the tests and its solution through the tecnique of *Locality Sensitive Hashing (LSH)*. Furthermore is shown as evaluating the predictions for a private attribute of a given user.

3.1 Collaborative Filtering and K-Nearest Neighbours approach

The *Collaborative Filtering (CF)* is a famous and popular recommendation algorithm many using in *Recommendation Systems*[1]. In general this class of

[1]is a family of algorithms that helps the people to effectuate several choices based on different aspects. For example Amazon use this family of algorithms to recommend several

methods is a system to filter the informations basing on the collaboration of several agents. The basic idea of this tecnique is to recommend items to users based on preferences and behaviors of other users in the system. The foundamental idea of this class of methods is that the preferences expressed by several users, can be aggregated and eleborated to provide a reasonable prediction for users that haven't declared preferences in the same system. Therefore this method analyzes relationships between users and interdependencies among products to identify new user-item associations.

The *Collaborative Filtering (CF)* provides several important approaches that can be sub-divided into three types of methods: *Memory based*, *Model based* and *Hybrid*.

The *Memory-based* approach consideres and memorizes the entire informations in a dataset. The principal two methods that compose it are known as: *Item-Based* and *User-Based* method.

The *Item- Based* algorithm returns recommendation evaluating the most similar items to those that a user has rated in his virtual history. Given a user this type of approach considers the items that this user has rated and computes how similar they are to the given item i and then selects k most similar items. In particular is evaluated a matrix having the following structure:

items.

	$Item1$	$Item2$	$Item3$	...	$Itemn$
$Item1$	sim_{11}	sim_{12}	sim_{13}	...	sim_{1n}
$Item2$	sim_{21}	sim_{22}	sim_{23}	...	sim_{2n}
$Item3$	sim_{31}	sim_{32}	sim_{33}	...	sim_{3n}
$Item4$	sim_{41}	sim_{42}	sim_{43}	...	sim_{4n}
...	...	...	...	...	...
$Itemm$	sim_{m1}	sim_{m2}	sim_{m3}	...	sim_{mn}

Figura 3.1: Item-based matrix

The similairity among items is evaluated through many several different mathematical formulations such as *Cosine based similarity*, *Person correlatio*n to name a few. After this phase the prediction is then computed by taking a weighted average of the target user's ratings on these similar items. The evaluation of similarity among items represents a critical step for the *Item-Based Collaborative Filtering* and it's very important to evaluate this value in a good way. An example of this method is used by Amazon that recommends to a final consumer possible items in his own shopping cart.
On the contrary, the *User- Based* approach considers the users present in the system and looking for a user the most similar users. This algorithm returns recommendations through a weighted combination of items of these users. Often this method is known as *k-Nearest Neighbours Collaborative Filtering*. For the inference of private attributes on Social Networks has been implemented the *Collaborative Filtering* algorithm with *User- Based* approach and the idea of this method has been adapted to the context of social platform. In this case the users of the social platform have a list of public attributes and basing on these public informations, the *Collaborative Filtering* algorithm researches for each user the most similar users present in the social

graph. These users are sorted for crescent similarity values and the first k represents the users with high value of similarity with the given user *(k- Nearest Neighbors approach),* known as *TopN.* The data provide from these users are elaborated to produce predictions for a private attribute of a given user, using a weighted combination of their eventually values for his own private attribute.

The *Collaborative Filtering* algorhtm, with *k- Nearest Neighbors* approach, is the following:

1. *for each user must be found users with similarity greater than 0;*
2. *select k users that have the highest similarity with the given user, usually called neighbours or TopN;*
3. *computing a prediction from a weighted combinations of the selected neighbors' values for the private attribute of the given user.*

The matrix that stores the similarity among users has the following structure:

	$User1$	$User2$	$User3$	...	$Usern$
$User1$	sim_{11}	sim_{12}	sim_{13}	...	sim_{1n}
$User2$	sim_{21}	sim_{22}	sim_{23}	...	sim_{2n}
$User3$	sim_{31}	sim_{32}	sim_{33}	...	sim_{3n}
$User4$	sim_{41}	sim_{42}	sim_{43}	...	sim_{4n}
...	...	...	...	...	...
$Usern$	sim_{n1}	sim_{n2}	sim_{n3}	...	sim_{nn}

Figura 3.2: User-based matrix

Also in this case there are many measures to evaluate the similarity among user, as will be shown in the next Chapters. The principal advantages of this

approach is the simplicity to implement it for any situation and the quality of predictions are rather good. The need to use the entire dataset is negative, in fact is the datasets stores many informations uses this data in memory isn't very easy and efficient. Another important aspect is the probability that some times this approach not make predictions for some type of users. This could occur when the user hasn't many items in common with other users of the same system and this point is very important because the similarity among users striclty depends from the number of common data that a given user has with the other people.

The *Model- based* approach involve a probabilistic model based on the dataset of ratings. The dataset is used to extract some informations, building a model to make recommendations, but isn't necessary to involve the entire dataset every time. For this reason this approach is different from the precedent and therefore present several benefits of scalability and speed. In fact the result models are much smaller than the entire dataset and then more efficient. Another different from the precedent approch is that the time required to query the model is often much smaller than to query the entire dataset. Although these advantages there are several problems such as the building of a model that is often time and resources consuming process. Another important problem is the utilization of the dataset, in fact since isn't use the entire dataset but only a part of informations, the predictions could not be accurate as for the *Memory-Based* approach.

Finally there is another method that is a combination of Memory-based and Model-based algorithms, known as Hybrid Collaborative Filtering. This last approach is important to avoid the limitations of the precedet methods the-

reby improve recommendation performance.

All the precedent algorithms can be sub-divided into two important parts: the similarity and the prediction phase. The first part is the pulsanting heart of the *Collaborative Filtering (CF)* because from the similarity values depend the results returned through the prediction phase.

The approach chosen for the purpose of this *Master Thesis* has been the *Memory Based* approach with *User Based* method. The reason is that the inference of private attributes is based on the publicly available data on a Social Network, therefore the importance to use all the public data is very important. This constraint is satisfied in the *Memory Based* approach that uses the entire available dataset and not only a part as in the *Model Based* approach. Another reason is the quality of predictions that with high probability is better in the first approach than in the second method. Obtain predictions that aren't approximate is important to understand the importance level of the public data on a *Social Network* for the inference analysis. The *User Based* method has been preferred than the *Item Based* method because is foundamental verify the importance of the users present in a Social Network. For example the predictions obtained using the relationship links and so the users present in this list, with high probability, are better than the case in which this link isn't considered. The reason of this result will be discuss in the next Chapters, but the principal motivation is that these type of users probably share with the given user tha same informations and interests.

3.2 Similarity Measures and their applications

In the data mining context a fundamental problem is to find similar items examining the available data of a system. For the inference of private attributes of the users on a Social Network, the problem is to find similar users, elaborating the public data shared by the users of the given system.

There are many and different metrics available to evaluate the similarity among items and for this specific problem have been used the *Jaccard Index* and the *Cosine Similarity* measures.

The *Jaccard Index*, commonly kown as *Jaccard Similarity Coefficient*, is a measure of the similairity between two sets of values. Given two sets of values A and B, this statistical index is defined as the ratio between the intersection and the union of the considered sets.

$$J(A,B) = \frac{A \cap B}{A \cup B} \tag{3.1}$$

In this specific case, the two sets A and B represent the public attributes of the users.

This coefficient return a value that belongs to the interval $[0,1]$ and the following cases are possible:

- $sim(A,B) = 0$, the given sets are different;
- $0 < sim(A,B) < 1$, the given sets are more or less similar;
- $sim(A,B) = 1$, the given sets are equals.

In figure 3.3 are shown two sets of elements A and B. Their intersection is equal to three and their union is equal to eight. The value of *Jaccard Index* is $\frac{3}{8}$.

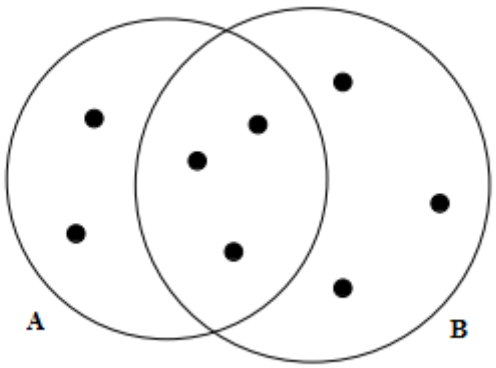

Figura 3.3: Jaccard Similarity example

The second metric of similarity is known as *Cosine Similarity Measure* and consists to evaluate the cosine of the angle between two vectors. This angle will be in the range 0 and 180 degrees, so no negative distances are possible. This measure indicates how related two vectors x and y, observing the angle between them. On following is shown the equation of this metric:

$$sim(x, y) = cos(\Theta) = \frac{x * y}{\parallel x \parallel * \parallel y \parallel} \tag{3.2}$$

For the precedent equation are possible the following cases:

- two vectors with an angle of 90° not share any attributes and they have a value of similarity equal to 0;

- two vectors with the same orientation have a value of similarity near to 1;
- two vectors with opposite orientation are very dissimilar.

For the *Cosine Similarity Measures* the list of public attributes has been elaborated as a binary vector. Given two list of attributes, the elements of the lists are joined in a unique list and, for every value in this new list, has been verified whether the considered values are present in the list of the given user. If the element is present in the list of a user, the corrispondent value in his own binary vector is 1, otherwise is 0.
Through these metrics, the users present in the Social Network can be compared for the similarity evaluation in two different ways:

1. comparing every possible pair of users;
2. comparing every user with only his own friends (importance of the friendship list).

The first way is not efficient from the computational time point of view. The reasons for this problem will be explained in the next section, introducing an important technique to solve this type of problem known as *Locality Sensitive Hashing (LSH)* with *MinHashing* technique.

3.3 Scalability Problems: Locality Sensitive Hashing (LSH) with Minhashing technique

The large amount of data that to be processed is very often an important problem that must be faced. In particular this problem is very recurring when the purpose of elaboration of the available data is to find similar documents in a high dimensional space and this number of documents is not very easy to manage and to elaborate. This point is very evident when these data must be compared to find, for each item, the most similar documents present in the given dataset. This type of elaboration entails many type of problems concerning the memory manage and the evaluation times that not always can be solved. Managing a lot of informations and comparing each possible pairs of documents means to use efficients data structures of memorization and high computational resources, but not always solve this type of problem. This problem there is also in the project of this Mster Thesis, in which the important and critical point is the comparison of the users to evaluate their own similarity value. When the number of social profile is elevated, as in this case, many data must to be compared, but not always this is possible. In particular considering n the number of user in the dataset, the computational cost for this evaluation is equal to $O(n^2)$. The size of the matrix that store the similarity values is equal to $n^2$and for this reason isn't easy to load it in secondary memory and evaluting its value.

To solve this problem has been implemented an impostant tachnique nown as *Locality Sensitive Hashing (LSH)* with *MinHashing* method. This algorithms efficiently tries to solve this important problem reducing the space of

high dimensional data and efficiently tries to find the most similar pair of documents (users).

The important idea of this technique is to reduce the size of the Similarity Matrix, obtaining a matrix of less size. To achieve this result will be used several hash functions to obtain an important matrix known as *Signature Matrix*. This important matrix will be submitted to several elaborations to obtain an important result: documents very similar efficiently will be hashed in the same buckets with high probability.

3.3.1 Minhashing

The presence of a high number of users and the elevate quantity of informations declared in the Social Networks represent an important problem for the memorization and the management of these informations. Suppose to represent this data in a matrix known as *Characteristic Matrix* , in which the columns represent the users and the rows represent the values of the attributes declared by these users of the social platform. The cell *(i, j)* of this matrix is equals to 1 if the value of the attribute i is declared by the considered user j, otherwise its value is 0. An example of this matrix is shown is the figure 3.4.

Element	*User 1*	*User 2*	*User 3*	...	*User n*
a	1	0	1	...	1
b	1	1	1	...	1
c	0	0	1	...	1
d	0	0	1	...	0
...	...	...	...	...	...
m	0	1	0	...	1

Figura 3.4: Example of characteristic matrix

The *Characteristic Matrix* with many probability has a very high dimension and for this reason its management is not very easy. Memorizing the data through this matrix is not efficient for the memorization of informations, in fact it take much space therefore it's only useful to visualize the data. Often this matrix is sparse, therefore it has many values equals to 0 than values equals to 1, so it wastes a lot of space and for this reason is recommended to store the data in another way.

The *Minhashing* tecnique tries to solve the precedent problem replacing the sets of stored values by much smaller representations called *signatures*. The equivalent matrix is known as *Signatures matrix* and the important idea is that similar users have similar *signatures*. In particular these new values are the result of a several elaborations known as *minhash* evaluated through the eklaboration of the *Characteristic Matrix.*

*T*his large number of calculations are obtained picks n random permutations of the rows of the matrix in the figure 3.4 . These n permutations h_1, h_2, ..., h_k allows to obtained the corrispondent minhash functions for a set S, therefore the corrispondent vector $[h_1(S), h_2(S), ..., h_k(S)]$ stored in the *Signature Matrix*. The *minhash value* $h_j(S)$ of any column is the number of the first

row, in the permuted order, in which the column has 1 as value.

An example of this type of matrix is shown in the figure 3.5 in which is drawn the *characteristic matrix* and a one of the possible permutation of its rows. The first column, which is the column for the *User 1*, has 1 in the row *a* and, for this reason, its corrispondent *minhash value* is *h(User 1)=a*. The second column for the *User 2* has 0 in the first three rows and it contains the value 1 in the fourth row, so its corrispondent *minhash value* is *h(User 2)=d*.

Element	*User 1*	*User 2*	*User 3*	*User 4*	*User 5*
a	1	0	0	1	1
b	1	0	0	0	0
c	0	0	0	0	1
d	0	1	1	0	1
e	0	0	0	0	1
f	1	0	0	1	0
...	...	...	...	...	...
m	...	...	...	...	...

Element	*User 1*	*User 2*	*User 3*	*User 4*	*User 5*
b	1	0	0	0	0
c	0	0	0	0	1
a	1	0	0	1	1
f	1	0	0	1	0
d	0	1	1	0	1
e	0	0	0	0	1
...	...	...	...	...	...
m	...	...	...	...	...

Figura 3.5: An example of permutations of the rows

The permutation of the rows of the *Characteristic Matrix* is not a functional way to obtained the *Signature Matrix*. In fact permuting the precedent matrix is possible in theory, but prohibitive in practice because of the number of

random permutations for millions of rows a high. This evaluation would request many computational time and the necessary sorting of the rows would take even more time.

To solve this problem in an efficient way, the k random permutations have been simulated through k random hash functions. These type of functions have been obtained using a universal hash family, shown in the equation 3.3 , that allows to evaluate k several random hash functions having the same structure.

$$h(x) = ((a.x) + b) \, mod N \tag{3.3}$$

where:

- a and b are random numbers belong to the range $[0, p-1]$with p prime number;
- N is a sizable prime number. This number must be sizable to limit the number of collisions between two different hash functions having the same imput.

Thus, instead to pick k random permutations of rows, are picked k random hash function $h_1, h_2, ..., h_k$on rows and for every user are evaluated a sets of min values obtained from the precedent hash functions.

Hash functions	*User 1*	*User 2*	*User 3*	.....	*User n*
Hash 1	$h_{1min}(1)$	$h_{1min}(2)$	$h_{1min}(3)$	...	$h_{1min}(4)$
Hash 2	$h_{2min}(1)$	$h_{2min}(2)$	$h_{2min}(3)$	...	$h_{2min}(4)$
Hash 3	$h_{3min}(1)$	$h_{3min}(2)$	$h_{3min}(3)$	...	$h_{3min}(4)$
Hash 4	$h_{4min}(1)$	$h_{4min}(2)$	$h_{4min}(3)$	...	$h_{4min}(4)$
...	...	...	...	...	...
Hash m	$h_{min}(1)$	$h_{min}(2)$	$h_{min}(3)$	...	$h_{min}(4)$

Figura 3.6: Signatures matrix

The matrix represented in the figure 3.6 can be obtained applying the following general algorithm:

Algorithm 3.1 General MinHashing algorithm

- Compute $h_1(r), h_2(r), ..., h_k(r)$.
- For every column j of the *characteristic matrix M:*
  - if $M(r, j) = 0$ do nothing.
  - else if $M(r, j) = 1$ then for each $i = 1, ..., k$ set $SIG(i, j)$ to the smaller value between $SIG(i, j)$ and $h_i(r)$.

Initially every value $SIG(i, j) = \infty$ and the result values represent the *Signature Matrix.* Clearly, as already mentioned, the representation of data through the *Characteristic Matrix* 3.4 is not an efficient way to store the data, and so the precedent algorithm must be adapted to the choisen data structure. Tha data structure used in this *Master Thesis* for the representation of the *Characteristic Matrix* is a *HashMap H,* where every users has got

a list of values declared for his own attributes and so the algorithm 3.1 has been adapting for this data structure.

Algorithm 3.2 Adapting MinHashing algorithm

- Compute a and b for each hash function $i \in [1, k]$.
- Set the value N.
- For every user $j \in H$ do:
  - consider his own list of declared values L.
  - for each $h_i \in h_1(x), h_2(x), ..., h_k(x)$:
    - for each $x \in L$:
      - compute $h_i(x)$;
      - set $SIG(i, j)$ to the smaller value between $SIG(i, j)$ and $h_i(x)$.

The principal two advantages of the signature matrix obtained from the algorithm 3.1 are the following:

1. the size of the *Signature Matrix* is smaller than the size of *Characteristic Matrix*. Infact both these matrix has the same number of columns c, but the *Signature Matrix* has a number of rows equal to the number of the considered hash function. Suppose that this number is equal to k while the number of values that an user can declared is m and with high probability $k \ll m$. For this reason the *Characteristic Matrix* has a dimension of $(m \cdot c)$, while the size of the *Signaure Matrix* is $(k \cdot c)$. This important property allows to store the *Signature Matrix* in an efficient way;

2. the computation of the hash function is very fast and easy, on the contrary of n random permutations of rows for the *Characteristic Matrix;*

3. The *Minhashing* tecnique preserves the similarity among two sets as shows in the equation 3.4 .In particular the probability that the minhash function for a random permutation of rows produces the same values for two sets equals the *Jaccard Similarity* of those sets. In other words, if are picked a random hash functions and have evaluated the minimum hashes for two sets, then the probability that they share the same minimum hash is equal to the ratio of their common elements to their total elements.

$$P(h_{min}(x) = h_{min}(y)) = J(x, y) \tag{3.4}$$

3.3.2 Signature matrix partitioning and user's mapping phase

Using the *Minhashing* tecnique is important to compress large documents into small *signature*, preserving the similarity among two or more documents. These properties are fundamental to give efficient the research of similar documents inside a high dimensional space, solving the problem to compare all possible pairs that may be impossible if there are too many elements in the given set. This is the starting point of the *Locality Sensitive Hashing (LSH)* algorithm that can be sub-divided in three principal phases.

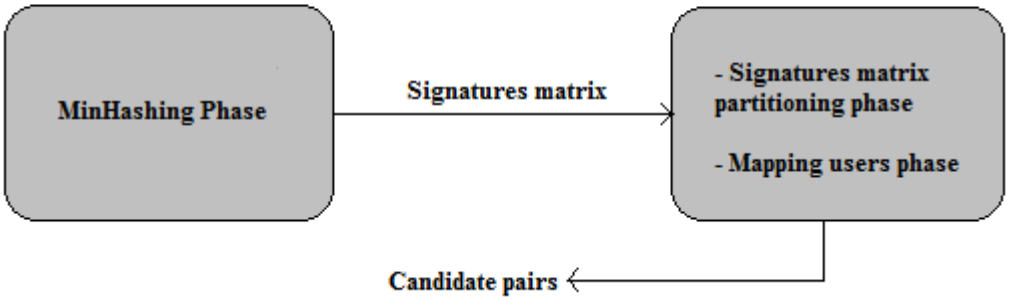

Figura 3.7: Locality Sensitive Hashing scheme

The purpose of *LSH* is to hash documents into different buckets several times, trusting that similar documents are more likely to be hashed into the same bucket that dissimilar documents are. The important point is that the documents hashed into the same bucket are *candidate pairs* to be compared with one of the precedent similarity metrics. The adavantage is that the algorithm compare only these candidates and not all possible pairs of the given dataset. The hope is that dissimilar items are not never hashed into the same bucket and therefore they will not never compare. Because of the presence of collisions could be possible that dissimilar documents are hashed into the same bucket, for this reason they are called *false positive*.
Obviously the hope is that the number of dissimilar documents, that are hashed into the same bucket, is a small fraction on the total data. In adding to this, the number of similar documents that are mapped into different buckets, and for this reason called *false negative*, should be a small fraction of the truly similar pairs.

As shown in the figure 3.7, Lsh is composed by three important parts:

- *MinHashing phase*, just described in the precedent section;
- *Partitioning phase*;
- *Mapping users phase.*

As earlier said, in the *MinHashing* phase the *Signatures Matrix* is built using a number of random hash function m chosen from a universal hash family. The number of these hash functions is important for the *Partitioning Phase,* in which the *Signature Matrix* (figure 3.6) is divided in b bands of r rows each such that $b \cdot r = m$. In other words the product among the number of bands and the number of rows for bands must be equals to the number of the chosen hash functions.

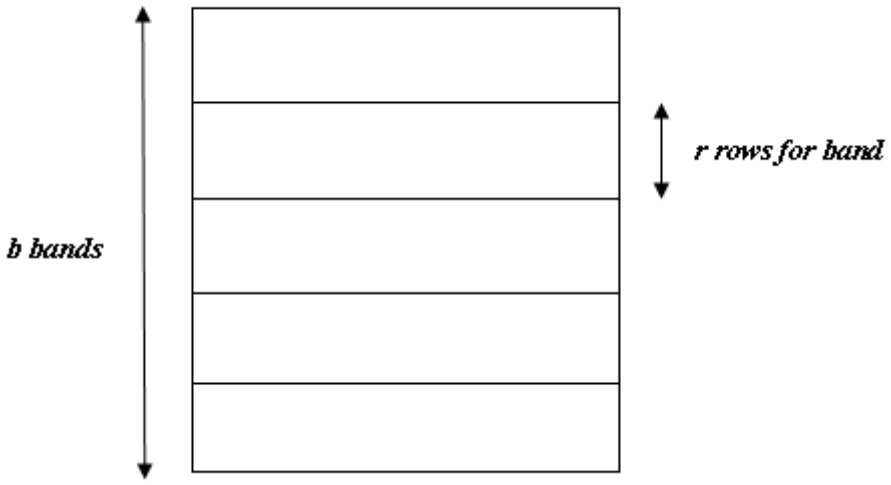

Figura 3.8: Signatures matrix partitioning

This subdivision is important because allows to hash a user into a bucket several times, therefore for every band. The need to have a division in several bands means for a user the possibility to be hashed b times (one for every *band*) in a bucket, where could be presented different users that are not necessarily equals for every bucket in which the user indeed has been mapped. In particular for each band there is a *hash function* that takes a vectors of r integers for every user (obtained through *Minhashing phase*) and hashed the given user into an array of buckets for the given band (*Mapping phase*) . This array of buckets is more known as *Hash Table*, that is a data structure where there is a corrispondence between a key and a value. In the figure 3.9 is shown an example of *Hash Table*.

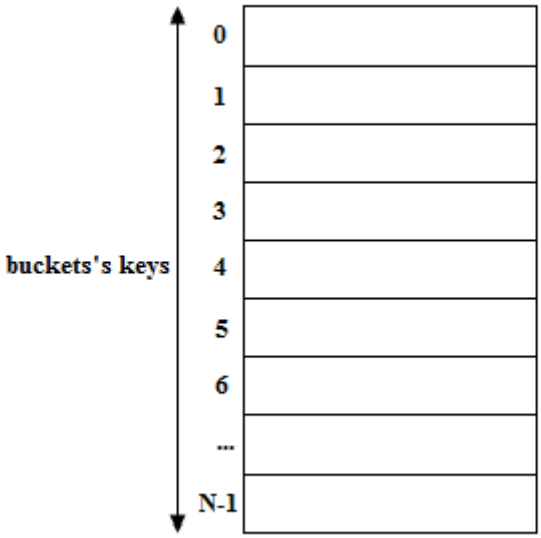

Figura 3.9: Example of hash table

In this case, the key represents the index of the bucket in which the user

will be hashed, while the values is the user itself. The key of this table is often obtained through a function called *Module,* that allows to control the dimension of the array in which the users should be mapped and it's represented by the following equation:

$$index(x) = x \mod N \tag{3.5}$$

The parameter x represents the input value and N represents the size of the *hash table* for each *band* that often is equal for every of these. The return value for this function is the rest of division, it is between $[0, N-1]$ and it represent the index of bucket in which an user will be hashed. Having the value of this function an inferior and superior boundary, different items could be mapped in the same bucket for a band because for different inputs could return the same value so creating a *collision* between different items.

When every user is hashed into a bucket of an *hash table* can be possible several scenarios for every band:

- some buckets are empty and so none user is hashed in this bucket;
- some buckets contain only one user;
- some buckets contains two or more users originating a *collision.*

The concept of *collision* in the *Locality Sensitive Hahing* technique is very important. For each band every user has an integer vector that give in input to a hash function that return an output value called *digest.* This

value is submitted to several elaborations and is very important because represents the index of bucket in which a user will be mapped. Two similar users likely will have the column vector identical for the most part of bands and with high probability they will hash in the same bucket. A good hash function guarantees that for the same input it return the same output, but although this property there is the problem of *false positive*, therefore when two dissimilar users share the same bucket.

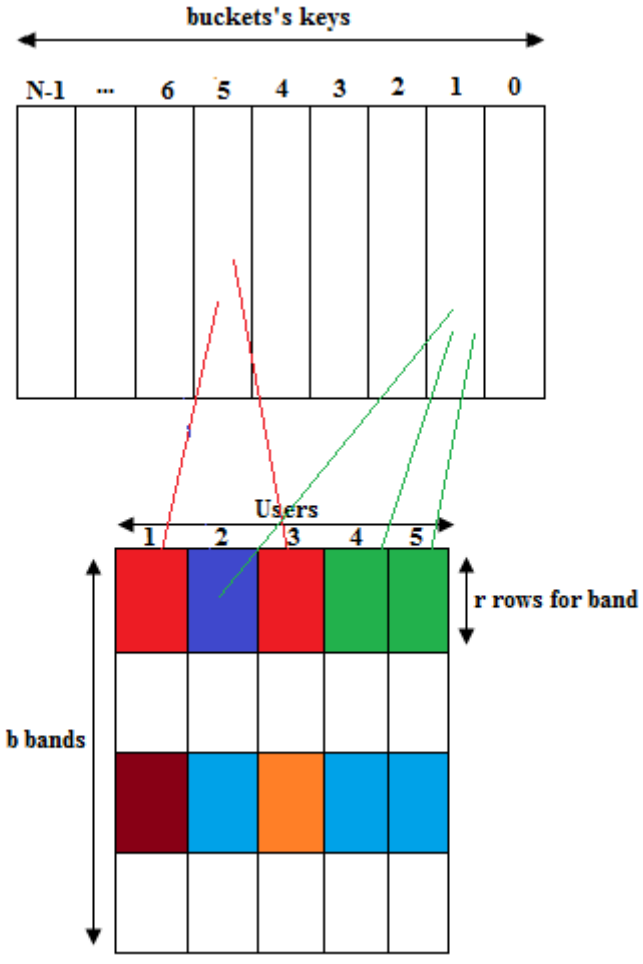

Figura 3.10: Example of mapping phase

In the figure 3.10 is shown an example of collisions for four users and four bands. In the first band the users one and three are hashed in the same bucket, but for the band three they mapped in different buckets. This means that these users are similar for the first band, but dissimilar for the third band. Obviously because of the collisions returned from the module function, could occurred that dissimilar users however are hashed in the same bucket. An example of this case occurs for the user two, four and five in the first band.

Two o more dissimilar users because of these collisions could be mapped in the same bucket and this case is not good. The users that share a bucket for a certain band are candidate pairs to be compared through one of precedent similarity metric. Wheter these users share the same bucket, but they are dissimilar, this represents an important problem. Is necessary monitoring these events reducing the possiblity that these occurring. For this purpose the principal ingredients that shuold be managed in a good way are two:

- the size of array for each band;
- resistent collision hash functions for the elaboration of integer vector that represent a user for a given *band*.

Controlling the size of array for each band means to manage the occurency of collisions and in particular their distribution in the buckets of a given array. First of all, in this specific case the size of all arrays is the same for each band. Choosing a number more or less elevated is very important because modifies the distribution of collisions in the several buckets. Often this size is

on the number of elements that must be hashed and therefore enough great, in particualar:

- if the size of this *Hash Table* is small, the probability that dissimilar users are hashed in the same bucket is very high. The paramter N of the *Module Function* return a value belonging to a small range and so the fraction of *false positive* notably increasing;
- if the size of this *Hash Table* is great, the probability that only similar users are mapped in the same bucket increasing and so the probability to have *false positive* is less.

3.4 Computing Predictions

As said in the precedent sections, the principal purpose of this Master Thesis is to infer private attributes of users through the public data available on a Social Network. The principal idea is to find for a user the most similar users and inferring his private attributes using the public informations available in the social platform. The strong point of this type of analysis, is the similarity among users that represents the principal base of *Collaborative Filtering (CF)* tecnique.

The first important step is to find for a user the list of his similar users and evaluating his own *Top N* users most similar with him. In particular every user has a list of users with their own similarity value and for every user this list is sorted in decreasing order. In this way the first values in the list are more near to the superior boundary of the possible admissible values. The

prediction of private attribute of a given user is evaluated on the base of values that his own similar users have declared for the considered attribute.

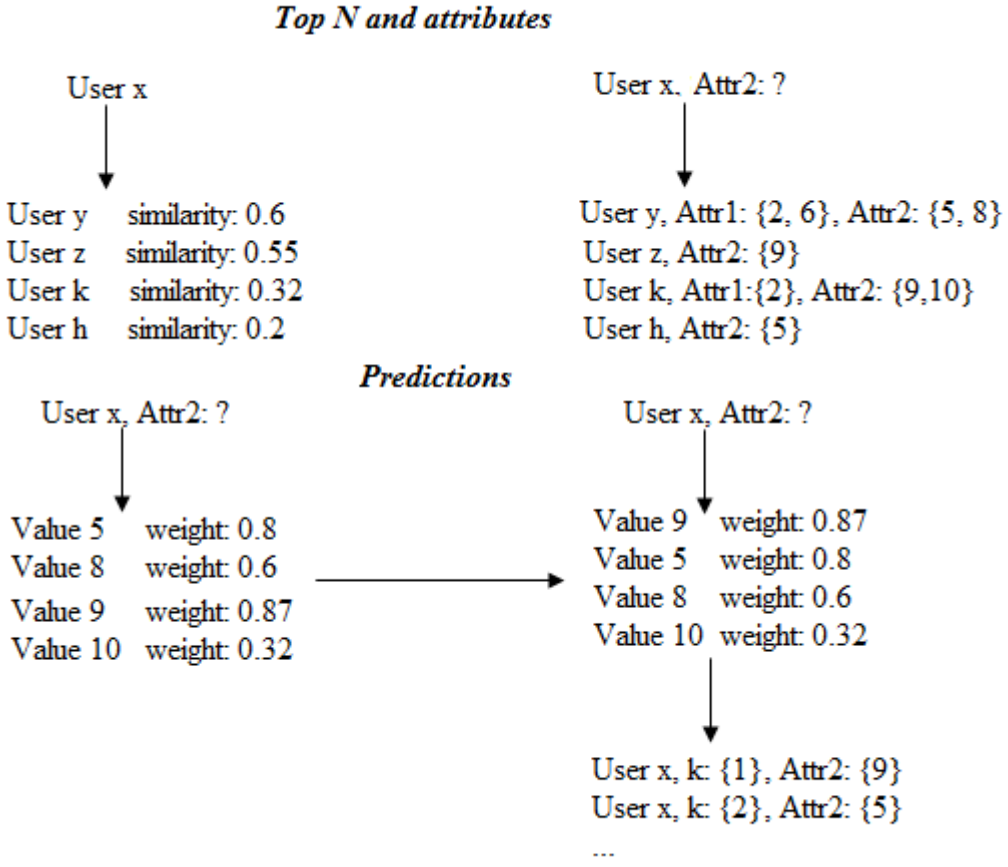

Figura 3.11: Example of weighted predictions

In particular the equal values are added on the base of similarity that these users has with the given user, therefore the predictions are weighted. After this elaboration these values have been sorted in crescent order and only the first k values have been considered for the inferring attributes. This parameter is important to control the candidate values considered for the private attribute that must be inferred. In the figure 4.4 is shown an example

of predictions in which the first part shows the *top* N users and their own attributes, while in the secondo part is described the prediction procedure. Obviously, the obtained list after the prediction of an attribute contains the most reliable values in the head of the list and the error prediction could increase considering values of k always greater.

Capitolo 4

Experimental evaluation

4.1 Target Dataset

The dataset used in the tests implemented in this project is obtained from the social plaform *Google+* [5]. In particular the dataset has been realized observing this *Social Network* for four months from June 2011 to October 2011, therefore when this *Social Network* has been launched with a invitation only test phase on June 2011. Every months corrisponds to a snapshot (0,1,2,3), but because of the great number of users collected during the four months, the tests for this Master Thesis are implemented only on the users obtained from the first snapshot and so from the crawler of June 2011. Initially the number of these users was more of 4,000,000, but because of the presence of users with less of two attributes, this number is decreased to 991545.
In general the dataset is organized as shown in the following figure:

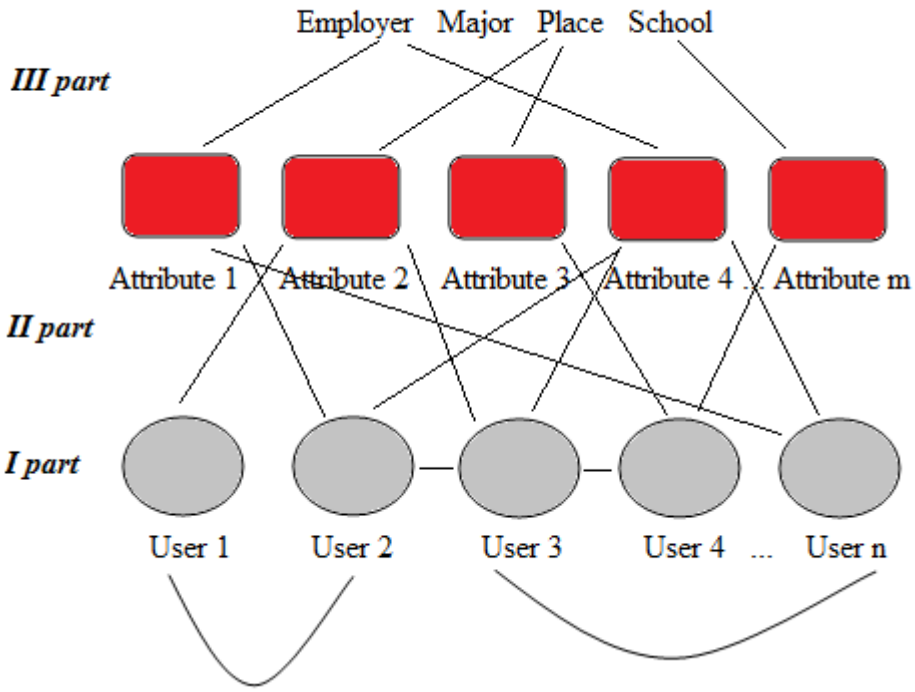

Figura 4.1: Dataset structure

- the first part represents the social graph and therefore the connection among users. The informations of this part are stored into a file in which every line is a direct link that connect a user with his own friends. The id of every users is an integer starting from 0 for the privacy policy.

$$UserFrom \quad UserTo \quad SnapshotId$$

- the second part represents the node attributes. The informations of this part are stored into a file in which every line of this file represent a user and his own attribute and the corrispondent snapshot in which

this user declares this value. The id of every attribute is a negative integer starting from -1.

$$UserId \quad AttributeId \quad SnapshotId$$

- the last part represents the type of attributes that could be: *Employer*, *Major*, *Place* and *School*. Every line of the file where this informations are stored is the following:

$$AttributeId \quad Type$$

Each user can express a list of values for every attributes and this point represent the principal motivation to introduce the parameter k for the prediction phase.
As shown in the next section, the informations stored in these file have been organized in a precise way to implement the tests necessary for the project.

4.2 Methodology

The important point of the project implemented in this *Master Thesis* is the dataset and in this case has been chosen the dataset of *Google +*. This dataset has been subdivided into two sets of data:

- *Training Set*

  - this part of informations has been used to evaluate the similarity among a pair of users that have declared at least one public attribute, therefore the users and their own public data.

- *Test Set*

  - this part of informations stores the attribute that should be inferred.

More precisely, the dataset has been subdivided into five *Folders* containing the same number of users and each folder is composed by two files:

- *File Base*

  - this file stores the users with their own public attributes. In particular are stored the users that have declared at least one public attribute. These files define the *Training Set* part.

- *File Test*

  - this file stores the users and their own private attribute, threfore the attributes that should be inferrred. These type of files define the *Test Set* part.

The private attributes of the users has been chosen as random, therefore whether a user has only attribute in his own list has been deleted from the system.
The final result has been evaluated as weighted mean of the results obtained by elaboration of five *Run.*

The similarity among the users has been used the *Training Set* adding one of *Test* file alternatively, therefore this file is different for every *Run*. This combination of data is necessary to guarantee that at least 20% of the attributes were inferred.

An example of management of precedent files has been shown in the following picture:

	Run 1	**Run 2**	...	**Run 5**
Training part	*File_Base_1*	*File_Base_1*		*File_Base_1*
	File_Base_2	*File_Base_2*		*File_Base_2*
	File_Base_3	*File_Base_3*		*File_Base_3*
	File_Base_4	*File_Base_4*		*File_Base_4*
	File_Base_5	*File_Base_5*		*File_Base_5*
	File_Test_1	*File_Test_2*		*File_Test_5*

Figura 4.2: Training Set

	Run1	**Run 2**	...	**Run 5**
Test part	*File_Test_2*	*File_Test_1*		*File_Test_1*
	File_Test_3	*File_Test_3*		*File_Test_2*
	File_Test_4	*File_Test_4*		*File_Test_3*
	File_Test_5	*File_Test_5*		*File_Test_4*

Figura 4.3: Test Set

Another important point are the parameter used in this project.

The similarity among users has been evaluated considering a pair of users in different way.

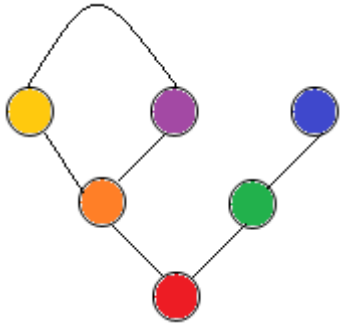

Figura 4.4: Example of social graph

On the base of the figure 4.4, have been implemented two methods to evaluate the similarity among users:

- a user has been compared with all of users of the Social Network. This means that the similarity has been evaluated also for users that aren't friends.

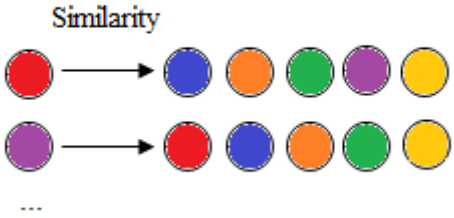

Figura 4.5: Example of similarity

- a user has been compared only with his own friends.

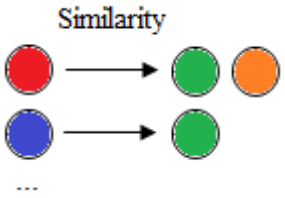

Figura 4.6: Example of similarity

For all the precedent methods have been used the *Jaccard Index* to evaluate the similarity among a pair of users. Really has been used also the *Cosine Similarity* measure but only for the case of similarity between a user and his own friends. The reason for this choice is that the results obtained for this metric have not been significant for the tests compared with the results obtained with the *Jaccard Index*.

After the evaluation of similarity, has been important to evaluate the number of *TopN* for every user, therefore the users more similar to him. This parameter have been set to a fixed value that is 1000, because the increasing or decreasing of this parameter wasn't useful for the results. More precisely:

- a small value meant to avoid users with important value of similarity;
- a high value meant to include users that had a insignificant similarity for the inference, threfore has been chosen an intermediate value to prevent this type of problems.

After the *similarity phase*, the next step has been to infer private attribute, controlling the number of possible values inferred for the given private attribute. The number of possible values has been set to a range $[1-7]$ and the reason for this superior boundary is that the *Precision* and *Recall* metric are

costant when k gradually increases.
As shown in the precedent sections, the method to evaluate the similarity for every pair of users has entailed important scalability problems and for this reason has been introduce the *Locality Sensitive Hashing (LSH)*. For this technique have been set the following parameters:

- the number of hash function is equal to 100;
- the number of possible bands, in which subdivided the *Signature Matrix*, has been evaluated by the formula $b * r = n$, where n is equal to the number of hash functions. As shown in the next section hasn't been possible to obtain result for the case greater than $b = 5$;
- the size of bucket's arrays have been set to the same value for each band. The reason for this choice is that the number of buckets in which the users are hashed in every band it must be equal for all the bands.

4.3 Precision and Recall

The results obtained from the implemented methods have been evaluated through two principal classifier *Precision and Recall*. These metrics are the basic measures used in evaluating search strategies and they are useful to evaluate the quality of the predictions.
Given a user afteer the prediction phase there are two sets of values: *Real* and *Predicted*. The first set of values represents the values that the user has really declared for his own attribute, while the second set of values is called *Predicted* and it represents the possible values that the user could declare for

the given attribute. These two sets are given in input to the *Precision* and *Recall* metrics. In particular:

- the *Precision* metric is a measure of results's relevancy and it represents the number of returned values that are real. It defined as shown at following:

$$Precision = \frac{(Real \bigcap Predicted)}{Predicted} \tag{4.1}$$

- the *Recall* metric represents the percentage of real values that are returned. It defined as shown at following:

$$Recall = \frac{(Real \bigcap Predicted)}{Real} \tag{4.2}$$

These two metrics are inverse therefore if the *Precision* curve increases, the *Recall* curve decreases and conversely.
These measures can be expressed in terms of false positive, false negative and true positive. In particular:

- true positive: results that are returns as true and they are real;
- false positive: results that are returns as true but they aren't real;
- false negative: results that are returns as negative but really they are true.

On the base of the precedent parameters, the *Precision* and the *Recall* formula become the following:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

A system with high recall but low precision returns many true positive and the number of false negative are few. This result is positive because the results returned from the system are for the majority exact. On the other hand a system with low recall and high precision has many false negative and few true positive.

The results shown in the following section have been obtained useing the *Jaccard Index* for the similarity evaluation.

4.4 Results

The purpose of this project is to show which among the submitted tecnique is the best to inference private attributes on a social patform useing the *Collaborative Filtering* tecnique. The results have been obtained from the ensuing tecniques:

- every user has been compared with every other user in the *Social Network*, so considering every possible pair of users;

- every user has been compared only with his friends, therefore considering the friend-relationship among users;
- *Locality Sensitive Hashing (LSH)* tecnique, that solving the scalability problems due to the great number of users signed up in the Social Network, therefore efficiently finding the most similar users of a given user avoiding to compare every possible pair of users.

4.4.1 Part I: Similarity among every possible pair of users

The results shown in this sub- section are obtained evaluating the similarity among every possible pair of users in the social graph. In other words, every user is compared with every other user of the dataset.

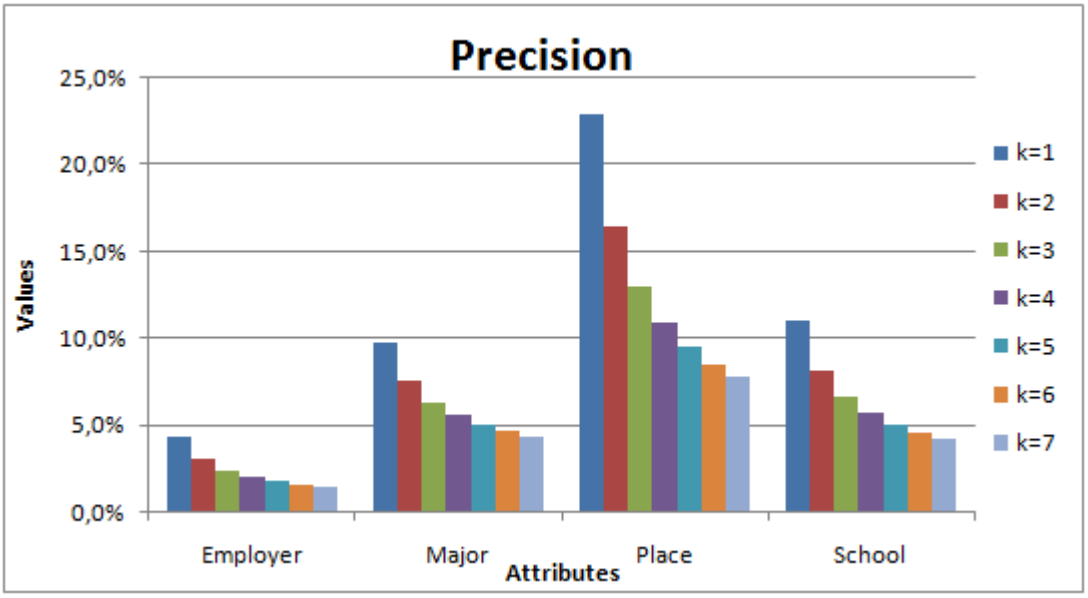

Figura 4.7: Precision

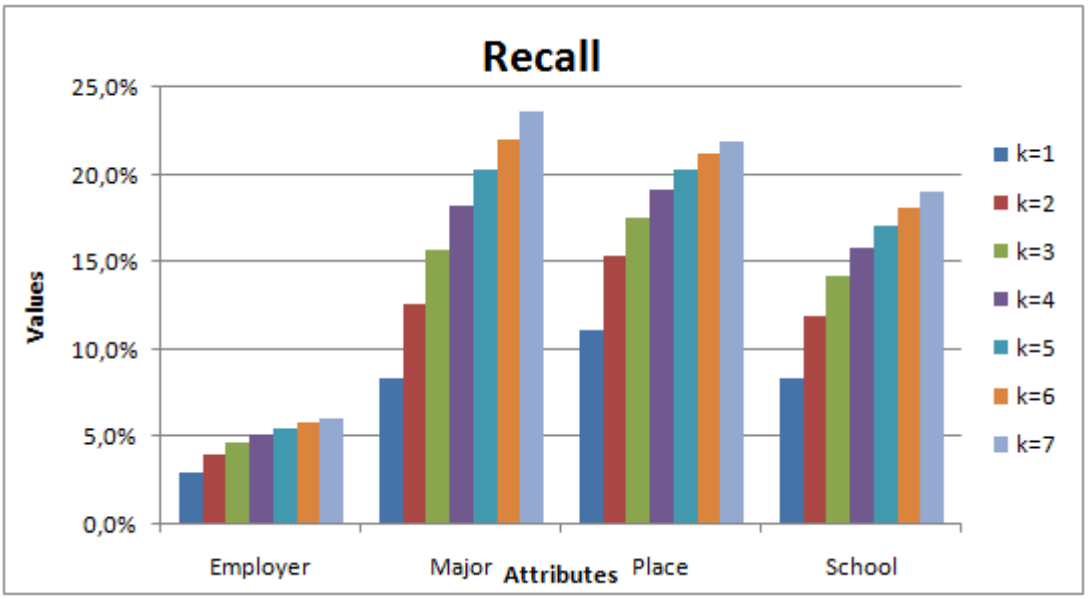

Figura 4.8: Recall

In the graphic 4.7 and 4.8 are shown the results of Precision and Recall respectively for the tests that evaluate the similarity among every possible pair of users.

In the graphic 4.7 is interesting to notice that the value of *Precision* decreasing with the increment of the parameter k. Is useful to remind that this parameter indicates how values are considered for the attribute that must be inferred and therefore considering an increasing value for k, means that the denominator of the formula 4.1 is great than the nominator of the fraction. Increasing k also means to risk that the considered value as results of the prediction are not equal to the real values for the considered attribute and therefore the formula 4.1 has a small nominator, a great denominator and so the result is not good. On the contrary the results for the Recall metric increasing because the intersections between real and predicted values is small

than the real values, ash shown the the formula 4.2.

Finally is shown the graphic that relate the *Precision* and *Recall* values.

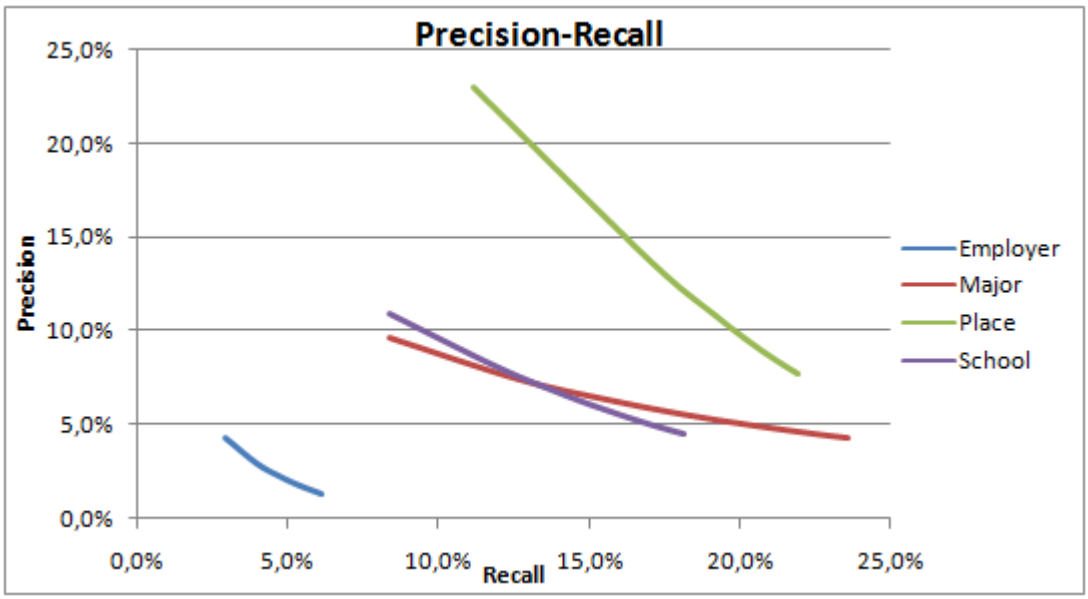

Figura 4.9: Precision- Recall

Is interesting to notice the difference of predictions for the four considered attributes. This difference for these results can be only explained from a theoric point of view, in particular the goodness of the results for an attribute depends by the public informations available for the attribute itself in the social platform and by the considered Social Network.

Having a lot of informations for an attribute means, with high probability, that a lot of users have declared this attribute on his own social profile and therefore considering for the similarity every possible pair of users increasing the probability to infer with a good result the given attribute. Obviously, comparing every possible pair of users present in the given dataset means to risk that a user is more similar with other users that not share his own value

for the inferred attribute. This happens because, through this method of comparing, a user is compared with users that he doesn't know sharing with them other attributes casually. Therefore is important to restrict the field of comparison among users, for example considering the friend relationship in the social graph.

The *Social Network* itself is very important to evaluate the value for the inference of an attribute of a given user. Earlier have been underlined the importance to have a lot of informations available for a given attribute that must be inferred for a given user and these point is strictly connected to the social platform. For example in a *Social Network* as *Google+* is not all of users share the information for the attribute employer, but for example in this type of platform many user declared the school attended and the place in which they live. On the contrary, a *Social Network* as *LinkedIn* with high probability contains many data for the attribute employer and probably the inference for it is more good.

4.4.2 Part II: friend relationships

In this section are shown the results obtained comparing a user with only his own list of friends. The important difference with the precedent tecnique is that a user is compared with users that probably share something with him and the size of this list is more less than before.

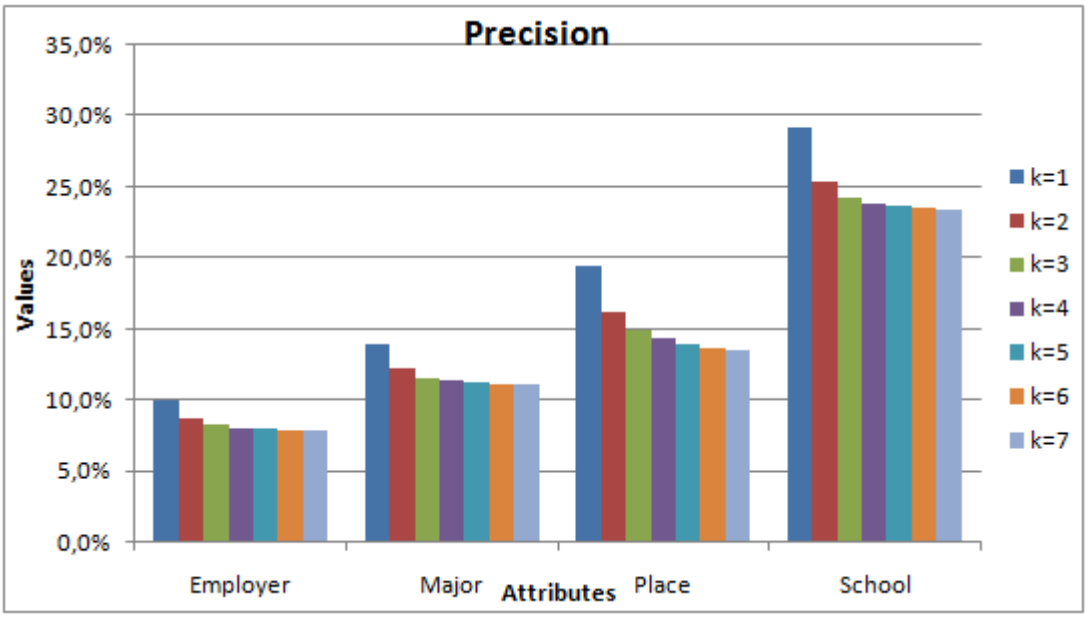

Figura 4.10: Precision

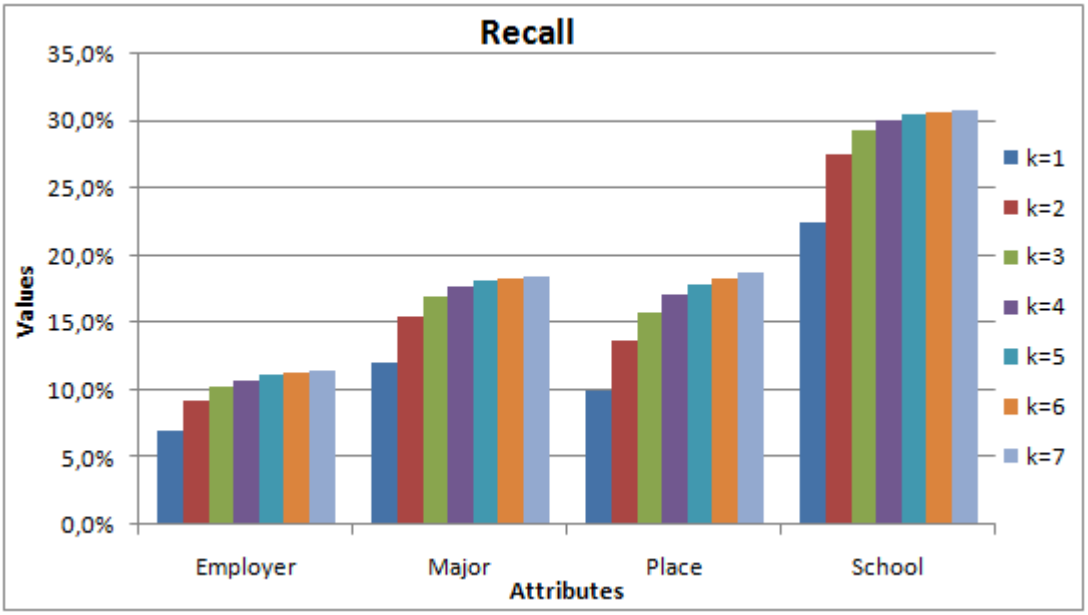

Figura 4.11: Recall

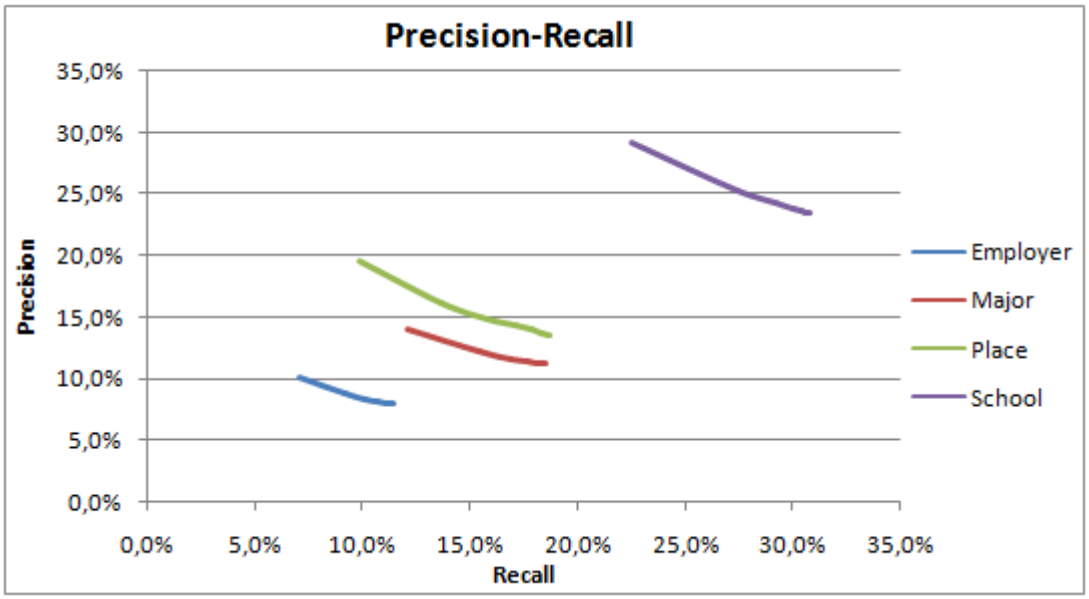

Figura 4.12: Precision- Recall

In the graphic 4.10, 4.11 and 4.12 are shown the result of *Precision* and *Recall* for the similarity of a user with his own friends.

In this case the quality of predictions is best for the attribute *School*. The reason for the improvement of the result about this attribute is that, with high probability, a user has in his own list of friends many users that share this attribute, in fact often a user is leverage to connect with people that share something with him. The probability that a user establish a relationship with his friends of school is very high and so the importance of public informations available for the inferred attribute are considerable also in this case. The graphic 4.13 reports the comparison of results for this method and the precedent. In particular the predictions are best for the method where a user is compared with only his own friends and this result is justified for the precedent reason.

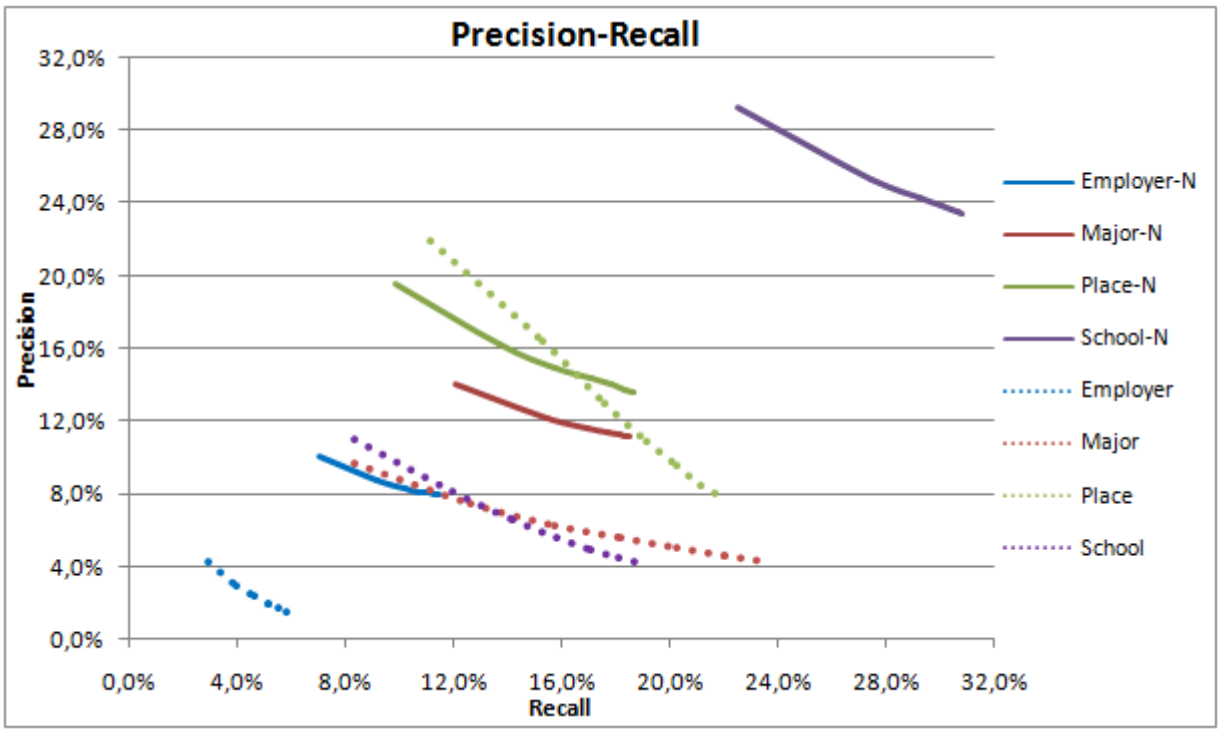

Figura 4.13: Neighbours vs Alls

4.4.3 Locality Sensitive Hashing

The scalability problems due to the similarity among every possible pair of users of the dataset has been solved through the introduction of the Locality Sensitive Hashing tecnique. For this purpose can be useful to remember the principal reasons for the introduction of this tecnique. The first reason is search for solve efficiently the problem to compare every possible pair of people for the similarity phase, that can be hard when there are many data to compare; the second reason is to improve the computational time for this evaluation because the comparison of all users take many time.

Is useful to recall the principal parts in which Lsh is composed:

- *Minhashing phase;*
- *Partinioning phase;*
- *Mapping phase.*

The Partitioning phase consists to divide the Minhash matrix into b bands of r rows each, so the given results have been obtained considering different values for the bands. The number of hash functions is equal to 100 and so the number of bands strictly depend from this value. In particular there are the following possible cases:

- $b = 1$, $r = 100$;
- $b = 2$, $r = 50$;
- $b = 4$, $r = 25$;
- $b = 5$, $r = 20$;
- $b = 10$, $r = 10$;
- $b = 20$, $r = 5$;
- $b = 25$, $r = 4$;
- $b = 50$, $r = 2$;
- $b = 100$, $r = 1$.

The cases with a number of bands greater than 5 have not been implemented because of memory problems. In fact, after the *mapping phase*, every user

has a list of the possible candidates used for the similarity comparation. This size of this list increase when the number of bands become great involving a great probability for a user to be hashed in many buckets with different users for every band.

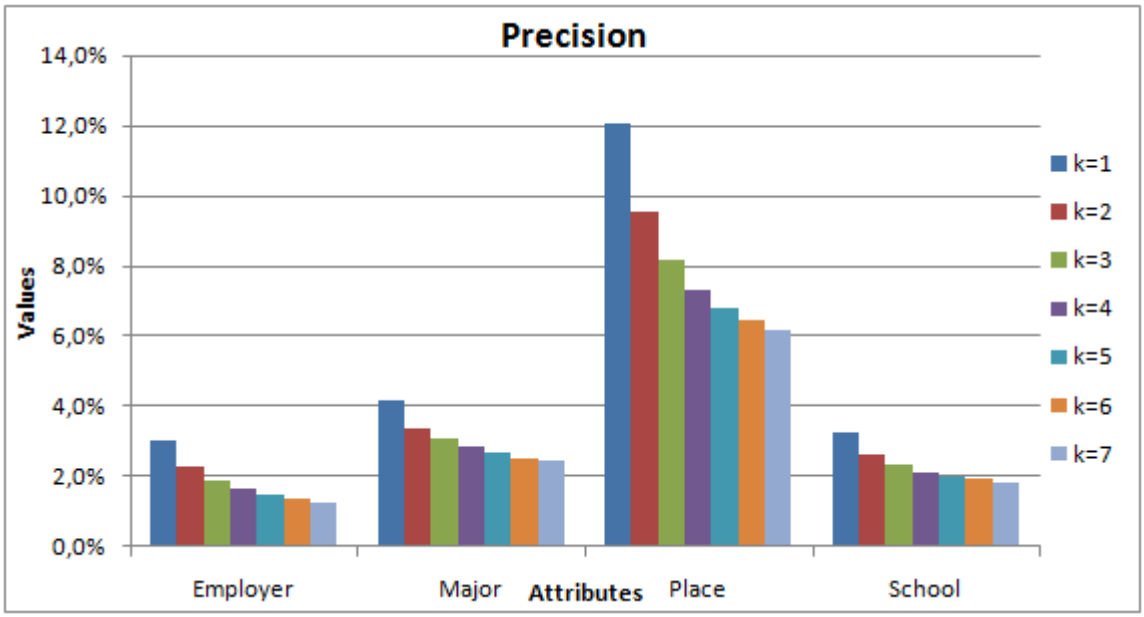

Figura 4.14: Precision for 5 bands

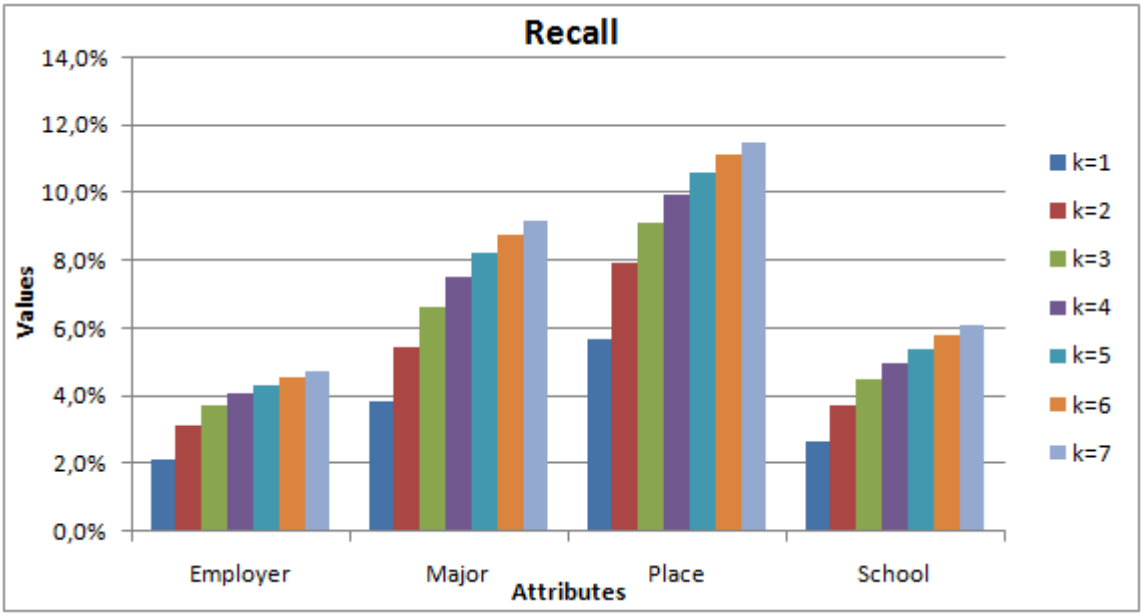

Figura 4.15: Recall for 5 bands

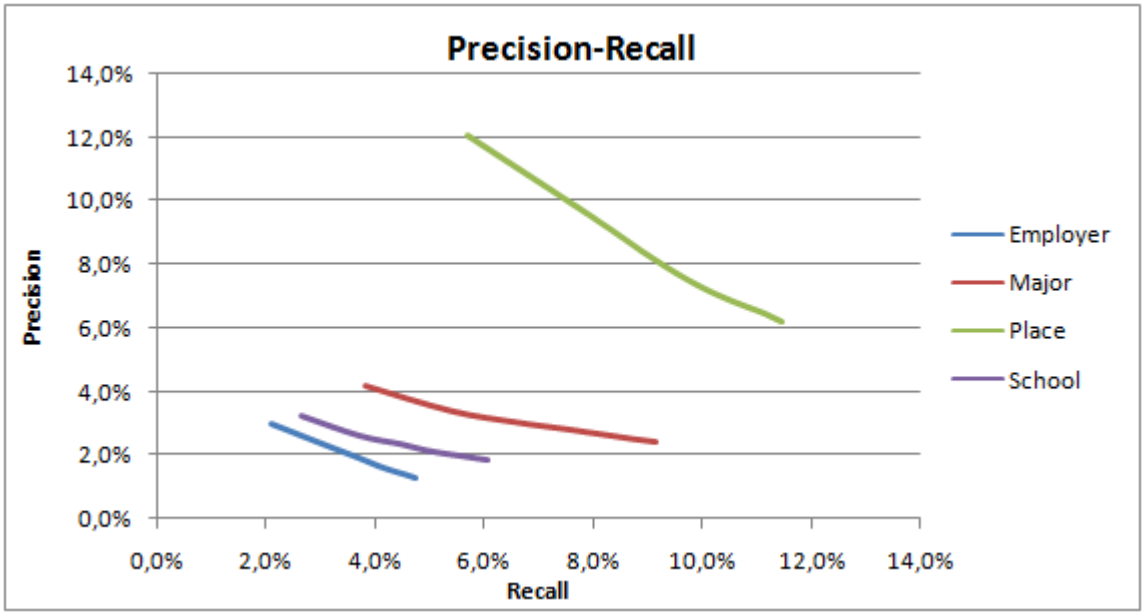

Figura 4.16: Precision- Recall for 5 bands

The graphic shown in the figures 4.14, 4.15 and 4.16 represent the results obtained for a number of bands equal to 5 and a number of rows for bands equal to 20. The better attribute that is inferred is *Place* as for the results obtained for the first method, so when every user is compared with all other users present in the Social Network. The important difference between these two cases are the quality of predictions that for the Lsh case is very low, as shown in the following graphic.

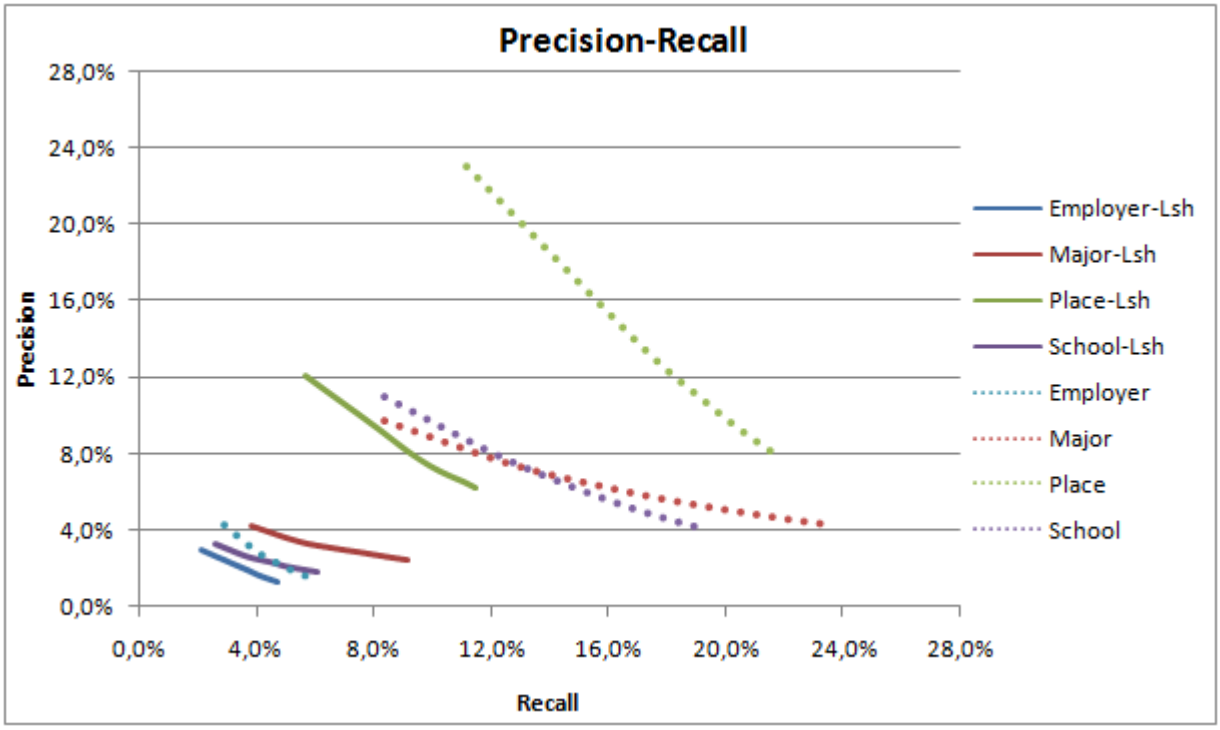

Figura 4.17: Lsh vs Alls

The results in the figure 4.17 show that the quality of predictions for the *Locality Sensitive Hashing (Lsh)* method is very low than for the other tecnique. This is very interesting for the analysis of the result because the quality of predictions is very low and this fact can seem very odd. Usually this tech-

nique is efficient and often returns good results, but not for this case. The reason of this fact is hidden in the contest in which this tecnique is implemented. The purpose of *Locality Sensitive Hashing (LSH)* for this project efficiently is to find the most similar users avoiding the comparison among every possible pair of users belonging to the given dataset. For the inferring of an hidden attribute is very important the list of similar users for a given user, because the predictions are implemented on the base of values that these users have declared for the considered attribute and for their own value of similarity with the given user. Having users with a value of similarity very high is not positive for the predictions but negative, in fact these users are not useful for the inference phase. For example, suppose to have two user with a high value of similarity, for example equal to 1. These users have the majority of the attributes equal, therefore the probability that a user has possible values for the hidden attribute is very low. The conclusion is that Lsh is an effient tecnique, in fact efficiently it finds the most similar users, but is not good for the inference of hidden attributes on a Social Network. The conclusion is that for the implementation of the *Inference Attack* are necessary similar users but not with a high value of similarity and therefore the *Locality Sensitive Hashing (LSH*) is a double- edged sword.
Finally as shown the graphics about the results obtained varying the number of bands:

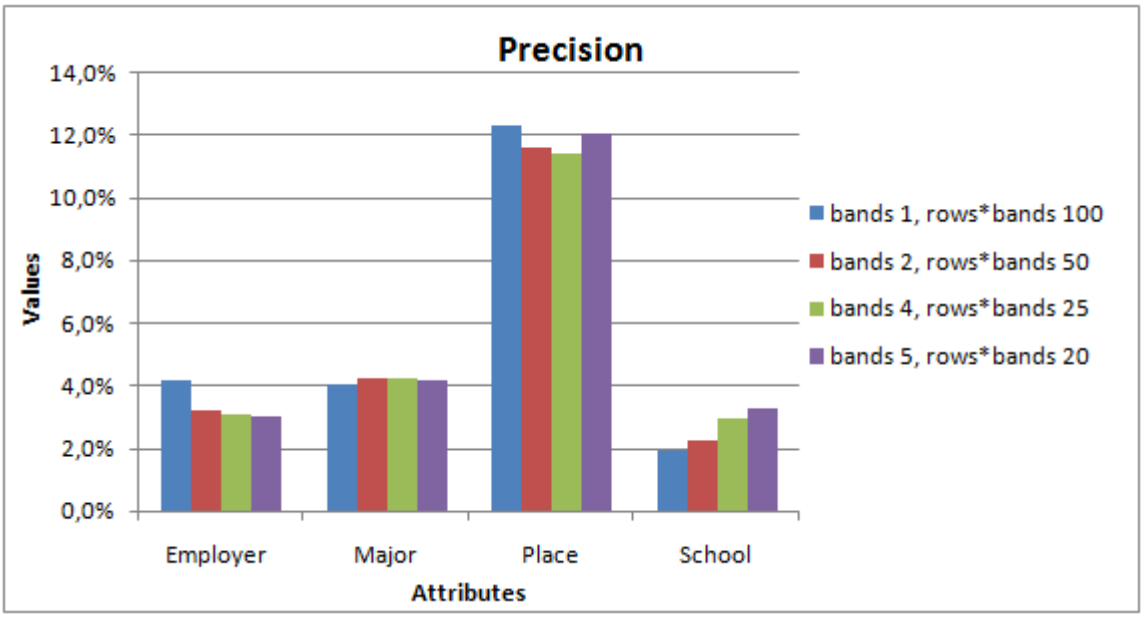

Figura 4.18: Precision varying the number of bands

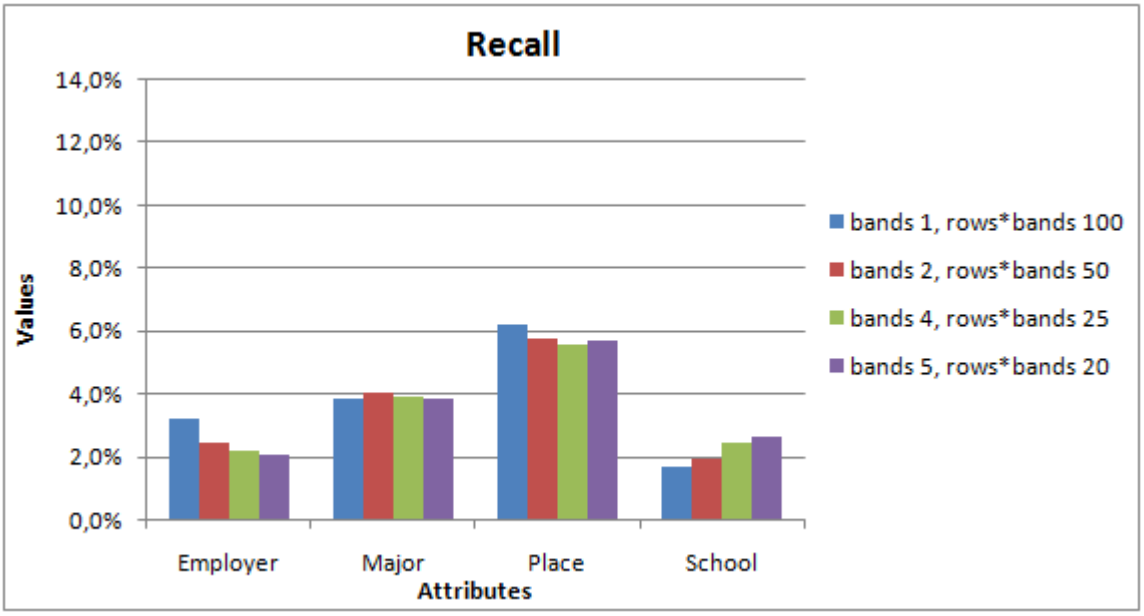

Figura 4.19: Recall varying the number of bands

The results for this variation of bands not improvement because increasing or decreasing the number of bands influences the probability of collision among users. Increasing the number of bands means that the probability that a user is hashed with many users in the same bucket is high and the size of his own list of candidates becomes great. Because of the collisions not all the users that share the same bucket could have a high value of similarity, so the users useful for the inferring of an attribute are more or less the same and for this reason the results are costant.

4.4.4 Total elapsed time

In the following graph the total time to evaluate the similarity among users for each one of the implemented methods is shown.

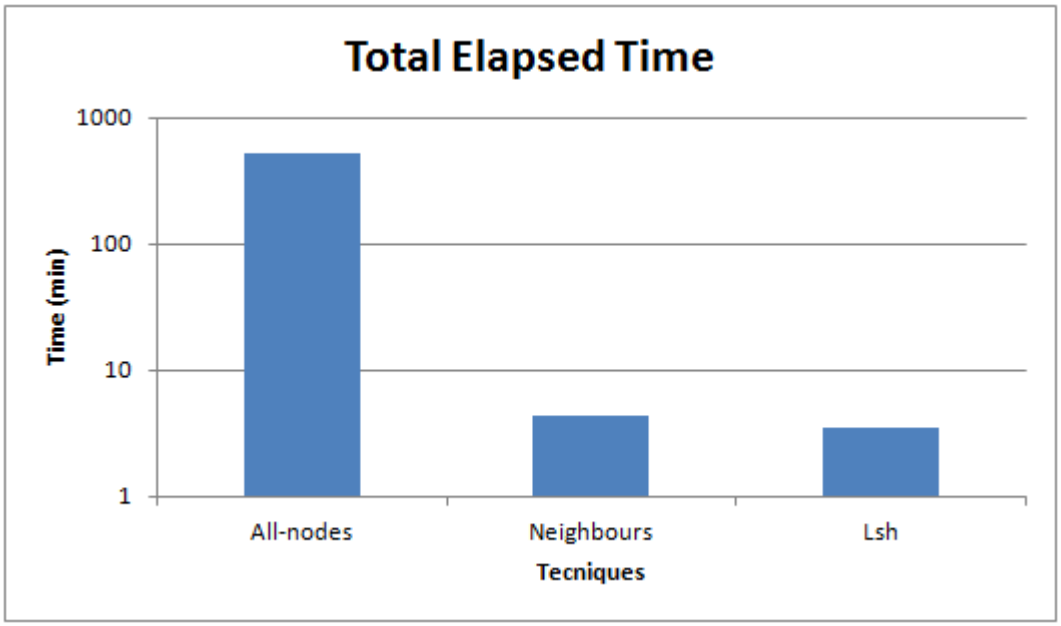

Figura 4.20: Total elapsed time

As explained in the *Collaborative Filtering (CF)* section, the Memory Based approach requires that the entire dataset is used to implement the intended analysis and this constraint is present in the method that compares every possible pair of users because every user must be considered for this first implemented method. The main consequences are an elevated computational time to evaluate the similarity among users and an inefficiency in terms of memory consumption to store the entire dataset. This problem isn't observed int the other two methods as they don't require to compare every possible pair of users in the dataset.

The *Neighbours* based algorithm only evaluates the similarity among a user and his own list of friends, therefore the time required for this comparison is less than the previous case.

The *Locality Sensitive Hashing (LSH)* algorithm shows a computational time similar to the *Neighbours* based algorithm because also in this case the comparison among every possible pair of users isn't necessary.

4.4.5 Cosine Similarity Measure

The following graphics show the principal results when the similarity has been evaluated among a user and only his own friends. For the remaining methods this metric hasn't been used because the results not entail important improvements.

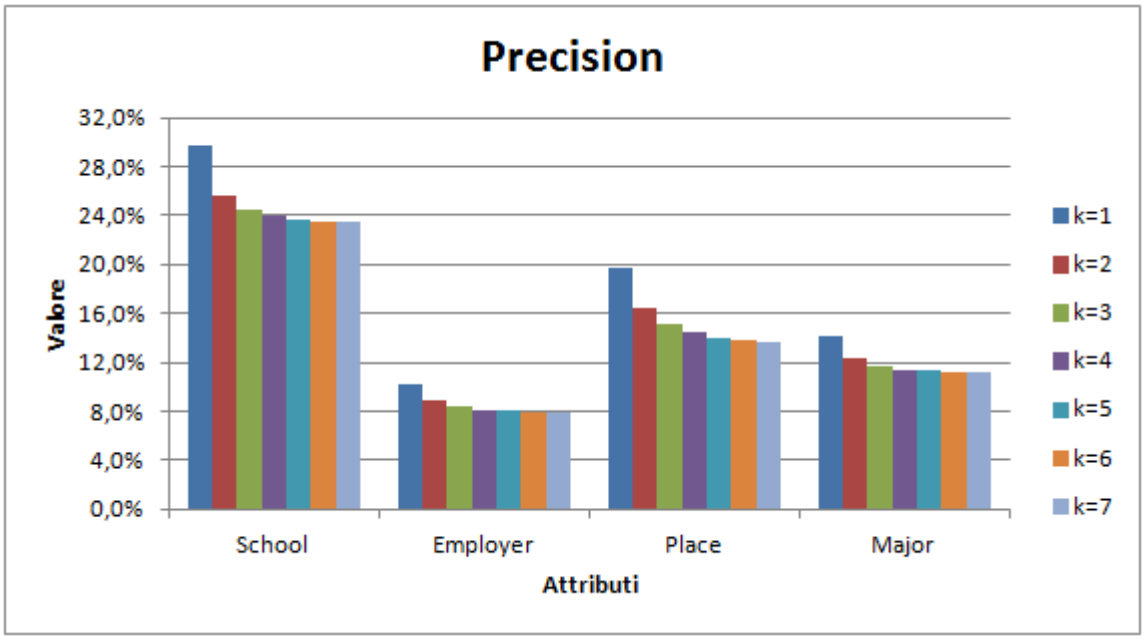

Figura 4.21: Precision

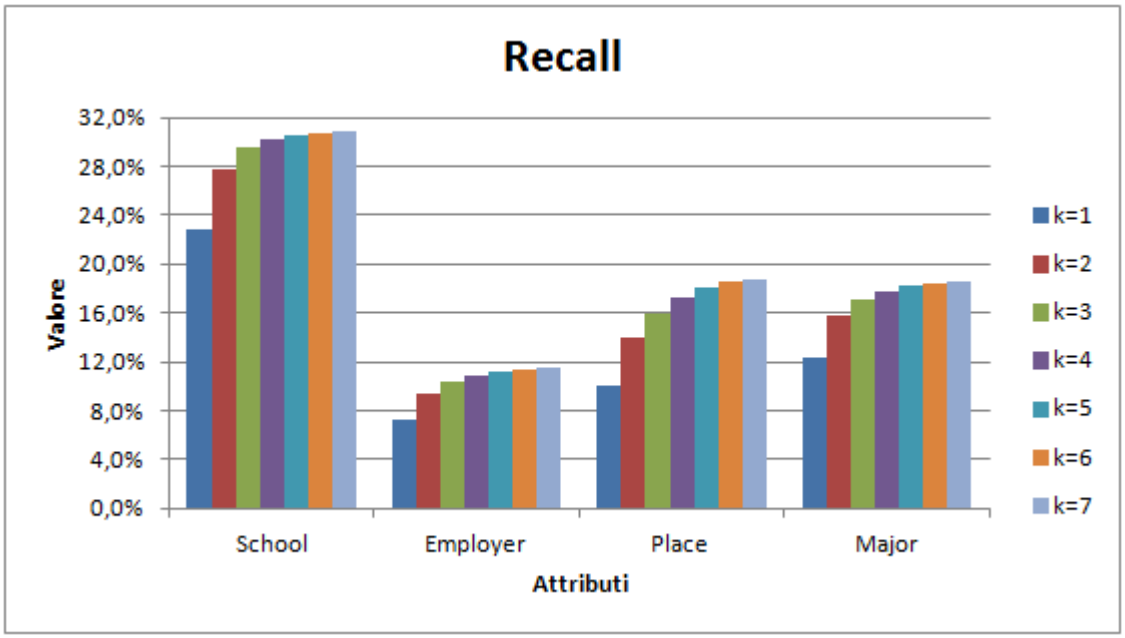

Figura 4.22: Recall

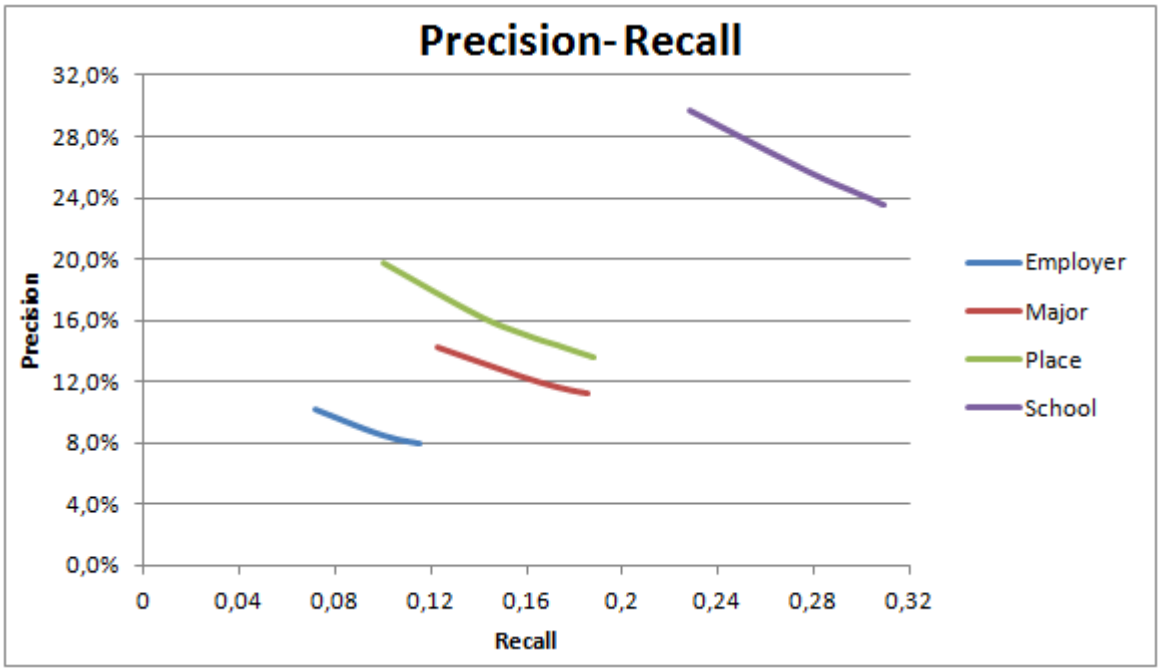

Figura 4.23: Precision- Recall

Is important to notice that choosing a different measure of similarity, the results are the same of the precedent case.

Capitolo 5

Conclusion

The Social Networks are becoming more common and popular among the people thanks to the variuos services they offer, such as the possibility to share many informations and stay connected with people you know. The sharing of data have highlighted an important issue: the management of the user's privacy.

Some people feel they must not hide some type of informations because in a first moment they may not seem important, but these informations leak a series of data that can potentially be exploited to infer private attributes that a user omits or with limited visibility.

The important question is the following: is possible to infer private attributes of a given user through the public informations available in the Social Networks?

The *State of Art* shows that it is possible to achieve good results and therefore that the informations shared on a social platform represent a seriuos threat to our privacy.

The methods proposed in this Master Thesis are based on a class of algorithms known as *Collaborative Filtering (CF),* based on the idea that users can "cooperate" releasing valuble informations for this type of analysis. In particular this means to evaluate the similarity among users, but in one of the implemented case have been observed scalability problems due to the prensence of many data. For this reason has been implemented the *Locality Sensitive Hashing (LSH)* tecnique that efficiently tries to solve this type of problem, reducing the dimensional space of this data.

The results obtained during the several tests are in agreement with the *State of Art*, showing that is possible to infer private attributes and that the quality of the predictions are not equal for all the attributes.This fact depends by several factors, such as the amount of informations available for each attribute, the *Social Network* chosen as dataset and the attribute on which the predictions are to be made. This last choice is linked to the *Social Network* selected as dataset, because the nature of some social platform favors the declaration of more data for some attributes compared than other.

Inferring private attributes is very important to sensitize the public opinion about the problem to preserve his own data and therefore preventing the divulgation of informations against his own will. The *Social Network* should realize possible solutions to prevent this type of analysis, helping the user to understand the possible way in which this is possible.

Bibliografia

[1] Elena Zheleva, LiseGetoor. *To Join or Not to Join: The Illusion of Privacy in Social Networks with Mixed Public and Private User Profiles.* Proceedings of the 18th international conference on World wide web, 2009.

[2] Chaabane, Abdelberi and Acs, Gergely and Kaafar, Mohamed Ali. *You are what you like! information leakage through users' interests.* Proceedings of the 19th Annual Network; Distributed System Security Symposium (NDSS), 2012.

[3] Mislove, Alan and Viswanath, Bimal and Gummadi, Krishna P and Druschel, Peter. *You Are Who You Know: Inferring User Profiles in Online Social Networks.* Proceedings of the third ACM international conference on Web search and data mining, 2010.

[4] Jure Leskovec, Anand Rajaraman, Jeff Ullman. *Mining of Massive Datasets. Chapter 3: Finding Similar items.*

[5] Neil Zhenqiang Gong, Wenchang Xu, Ling Huang, Prateek Mittal, Emil Stefanov, Vyas Sekar, Dawn Song. *Evolution of Social-Attribute*

Networks: Measurements, Modeling, and Implications using Google+. ACM/USENIX Internet Measurement Conference (IMC), 2012.

[6] Jianming He, Wesley W. Chu. *Protecting Private Information in Online Social Networks.* Computer Science Department, University of California, USA.

[7] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. *Item-Based Collaborative Filtering Recommendation Algorithms.*

[8] Patel Madhuri. *Different sanitization techniques to prevent inference attacks on social network data.* International Journal of Computer Science & Engineering Technology.

[9] Jack Lindamood, Raymond Heatherly, Bhavani Thuraisingham and Murat Katarcioglu. *Inferring private information using social network data.*

[10] Justin Becker, Hao Chan. *Measuring privacy risk in online social networks.*

[11] Raymond Heartherly, Murat Kantarcioglu and Bhavani Thuraisingham. *Preventing private information inference attacks on Social Networks.*