POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

Automatic alignment of user identities in heterogeneous social networks

Relatore: Piero Fraternali
Corelatore: Alessandro Bozzon

Tesi di Laurea di:
Giorgio Sironi, matricola 764852

Anno Accademico 2011-2012

Abstract

Record linkage is a well-known task that attempts to link different representations of the same entity, who happens to be duplicated inside a database; in particular, identity reconciliation is a subfield of record linkage that attempts to connect multiple records belonging to the same person. This work faces the problem in the context of online social networks, with the goal of linking profiles of different online platforms.

This work evaluates several machine learning techniques where domain-specific distances are employed (e.g. decision trees and support vector machines). In addition, we evaluate the influence of several post-processing techniques such as breakup of large connected components and of users containing conflicting profiles.

The evaluation has been performed on 2 datasets gathered from Facebook, Twitter and LinkedIn, for a total of 34,000 profiles and 2200 real users having more than one profile in the dataset. Precision and recall are in the range of cross-validated 90% depending on the model used, and decision trees are discovered as the most accurate classifier.

Sommario

Il record linkage è un problema noto che si propone di collegare differenti rappresentazioni di una stessa entità, che risultano essere duplicate in una base di dati; in particolare, la riconciliazione di identità ne è una specializzazione che si propone di riunire differenti record rappresentanti la stessa persona. Questo lavoro affronta il problema nel contesto dei profili utente dei moderni social network. L'obiettivo perseguito è l'allineamento dei profili di diverse piattaforme, ovvero il collegamento di tutti e soli i profili della stessa persona su piattaforme differenti.

Questa tesi valuta e confronta alcune tecniche di apprendimento automatico per le quali sono state definite distanze speficiche per il problema (e.g. alberi di decisione e macchine a vettori di supporto). In aggiunta, viene valutata l'influenza di tecniche di post-processing come la separazione di componenti connesse e di utenti contenenti profili in conflitto fra loro.

La validazione delle tecniche è stata eseguita sperimentalmente su due dataset estratti da Facebook, Twitter e LinkedIn, per un totale di 34,000 profili e 2200 persone aventi più di un profilo. Precisione e recall sono a livelli di 90% in cross-validazione a seconda del modello utilizzato, e gli alberi di decisione sono designati come il classificatore più accurato.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank Alessandro Bozzon and professor Piero Fraternali for directing this work with regular feedback, one of my preferred Lean tools.

I also recognize that the software developed for this academic work has benefited from the teachings of the Extreme Programming User Group in Milan.

Dedication

Ai miei genitori che mi hanno supportato in anni di studi,
a Kiki che sa quanto impegnativo è il lavoro dietro una tesi,
ai ragazzi dell'oratorio di San Crispino

1 Introduction

In the last years, online social networks such as Facebook, Twitter and LinkedIn have reached a formidable penetration everywhere Internet access is available. As such, these networks are the repository of a vast amount of knowledge about people, their relationships with each other, their areas of interests, and so on. There has been a vast amount of studies and commercial developments on the kinds of information that can be extracted from a person's online presence or from the APIs of the services that he uses.

However, each information extraction scenario is currently limited to a single platform: a person can be extracted from a network such as Facebook and his details can be analyzed, but the results are normally not put into correlation with the information extracted from other platforms.

In order to obtain a more complete understanding of the interests and capabilities of a person, the information from the different networks should be aligned, and an holistic analysis should be performed. Unfortunately, such an alignment requires the exact identification of people through social networks, a task hindered by the access limitation of social networks, and by the identity camouflaging or dissimulation techniques (e.g. nicknames, private fields, fake avatars) adopted by the users.

The problem addressed by this thesis is the linkage of online social networks profiles, spanning from the retrieval of entities set from multiple platforms, to their reconciliation as belonging to the same conceptual person (which is identified by the set of profiles linked together and not by additional information.)

More formally, in this domain:

- **platform** is the denomination for online social networks, such as Facebook and Twitter.
- **Entities** are profiles extracted from a platforms.
- The term **pair** is intended as a pair of profiles.
- A **person** or **user** is identified as a set of matched profiles, usually from 1 to N where N is the number of platforms.

1.1 Relevance

The importance of the problem is related to the presence of subsequent tasks of analysis or model learning whose accuracy can be improved with data extracted from different accounts of the same person.

For example, the *expert finding* task attempts to select a set of experts that are knowledgeable on a particular topic; by leveraging the information of multiple profiles of the same person expert finding algorithms have more data to work with. At the very least, duplicate results generated by finding the same expert's accounts on multiple platforms can easily be eliminated as they are recognized as a single person.

Along with expert finding, other tasks thay may take advantage from data from multiple profiles of a user are:

- The research of scientific collaborators, which for example in Schleyer et al. [30] sees the ability to exploit social networks and the aggregation of data from different sources as a requirement in expertise location.
- Crowdsourcing and expertise retrieval, which contain a component of identity reconciliation where we want to correctly consider all documents and material pointing to a candidate expert as a single score instead of to multiple fictitious people. Conversely, we would want to distinguish between homonyms when assigning the same scores (Balog et al. [2]).
- Customization opportunities, that could mitigate the cold start problem (for example in Carmagnola and Cena [5]), as new platforms cannot provide a

tailored user experience or recommendations until they gather data from their users. Correlating profiles with the ones of other platforms is a better starting point than relying on an empty profile.

- The study of individual personality and its relation to online behavior, that can use record linkage between survey-based and social-network data as a preliminary step (Wehrli [34]).
- Brand reputation analysis, which may track which users spread content from one network to another.

In general, analyses that span multiple domains are also an interesting field for profile alignment as users tend to post about different topics on different platforms. For example, LinkedIn is widely viewed as a professional network while Facebook as a personal one: linking friends from both Facebook and LinkedIn could at the very least rank the importance of work contacts, or even find common hobbies with our work colleagues.

1.2 Domain-specific issues

The problem of linking multiple representations of the same person is called *record linkage* (or *duplicate detection*, or *identity reconciliation*) and has a vast literature spanning from probabilistic to machine learning techniques.

However, there are several issues related to the domain of online social networks that are more specific than general person-based linkage:

- Anagraphical values are not always of high quality. It's common to use not only names but also informal abbreviations and pseudonyms, which would never be found on census or company data.
- There is an undetermined quantity of missing fields due to omittance (only a few fields like the name are mandatory, while birth dates and current locations are not always available) and because of restricted permissions and privacy reasons. Not all of the personal details of a profile are always available as the full database is private and belongs to the platform's.
- There is additional information about pairs and their compatibility. There is the possibility of conflict between users: two profiles with different platform-

wide autogenerated IDs, but belonging to the same platform are highly unlikely to represent the same person (they would correspond to duplicate Facebook or LinkedIn accounts).

- Non anagraphic fields can be considered: for instance, information about the connections of profiles (such as friendships and followships) can in principle carry information about their identity.

1.3 Original contribution

The setting of this thesis is the application of machine learning to the alignment of user identities, in the context of online social networks. Several models are trained and evaluated to be used to determine if pairs of profiles from different social networks belong to the same person or to different ones.

With respect to the state of the art, this work's contributions are:

1. The characterization of domain-specific features for classifying pairs of profiles that are mostly applicable to pairs of profiles extracted from an online social network, with all the peculiarities explained in the last section.

2. The production of a dataset containing ground truth in the order of thousands of profiles to evaluate the results of different learning algorithms.

3. The evaluation on the dataset of multiple existing approaches such as hierarchical clustering, decision tree induction and support vector machines.

The implementation of these techniques consists of an application which is able to extract sets of aligned profiles from Facebook, Twitter and LinkedIn by starting from a few seeds, users that have willingly authorized the application to access their accounts through the standard social network APIs. The number of seeds required is several orders of magnitude inferior to the number of users obtained during the extraction.

1.4 Thesis organization

The remainder of the document is organized as follows.

Chapter 2 first explores the prior art in the field of record linkage, with a look to the literature describing usage of unsupervised and supervised learning. The base record linkage problem is defined in mathematical terms along with several already existing solutions.

Chapter 3 characterizes the different classifiers under test and the post-processing variations that can be applied to their result. The design of the application and its integration with the online platforms is discussed.

Chapter 4 describes the evaluation of the different models over a real dataset. The statistical significance of the differences in performance between models is investigated, and several non-functional aspects are discussed.

Finally, chapter 5 summarizes the findings of this work and recollects some ideas for future research in the field under study.

2 Background

2.1 Problem definition and model

The record linkage problem is also known as the problem of *duplicate detection* or of *entity reconciliation*. In this context, representations of a person are the subject of study and the problem itself can be called *people reconciliation*. The rest of this section will always refer to:

- *entity* to denote a record, a particular representation of a person like his anagraphic information in the university database, or his own profile on Facebook.
- *Person* as a single physical person, conceptually spanning all the entities that represent her in the available databases. A person may be composed of 1 to N entities.
- *Pair* to denote a couple of entities that may or may not pertain to the same person.
- *Field* to denote an attribute of an entity, like its name or birth date.

This terminology is biased towards the people-based subfield of record linkage, but helps grounding the concepts explained in this background section and is in tune with the scope of this thesis.

In the standard mathematical model from Fellegi and Sunter [11], each person may belong to two populations A and B, whose elements can be denoted by a and

b respectively. The assumption is that some elements are common to the A and B set, but they are likely to be represented differently or incompletely in the different sets.

The set of ordered pairs is defined as:

$$A \times B = \{(a,b); a \in A, b \in B\}$$

and as such contains all the possible pairings between elements of A and B. The taks of people reconciliation is then to classify each pair in this set as belonging either to the *matched* or to the *unmatched* sets:

$$M = \{(a,b); a = b, a \in A, b \in B\}$$

$$U = \{(a,b); a \neq b, a \in A, b \in B\}$$

where $=$ is an equivalence relationship. We can distinguish between the conceptual people represented in the sets, elements a and b, and their recorded information, $\alpha(a)$ and $\beta(b)$.

Then, a and b are the same for each pair in M but $\alpha(a)$ and $\beta(b)$ may differ slightly. For example, the name of a person can be written as *FirstName LastName* in $\alpha(a)$ but as *LastName FirstName* in $\beta(b)$; or an abbrevation may be used for one version of the name, such as *Alex* instead of *Alexander*; or a pseudonym may be used instead of the real name, rendering the comparison between the name fields moot.

Conversely, $\alpha(a)$ and $\beta(b)$ can be equal (or very similar) even when the pair is in U, in case of homonymy; many different database records of *John Smith* or *Luca Rossi* exist without necessarily representing the same person.

The model does not exclude that A can be the same set as B: in this case the problem assumes the name of *duplicate detection* as the goal is to find duplicates a and b of any person that is represented multiple times in the list.

The result of the comparison of records a and b is called the *comparison vector*, and each element of this vector is a function of the whole $\alpha(b) and \beta(b)$:

$$\gamma[\alpha(a), \beta(b)] = \gamma_1[\alpha(a), \beta(b)], ..., \gamma_k[\alpha(a), \beta(b)]$$

although in most of the cases, when $\alpha(a)$ and $\beta(b)$ are database records, it is just a function of one of their fields.

The set of all possibile realizations of γ (a function on $A \times B$) is called *comparison space*, and if the cardinalities of A and B are comparable grows quadratically in size.

This model is generally applicable for all record linkage specializations, while alternative approaches are possible for the actual classification task. The approaches range from probabilistic to supervised ones.

The models also implies no loss of generality in considering just two populations A and B, as for additional populations such as C can be considered in isolation for linkage with A and B respectively; D for linkage with A, B and C; and so on. The M and U sets are the union of the respective pairwise matched and unmatched sets:

$$M = M_{ab} \cup M_{bc} \cup M_{ac}$$

$$U = U_{ab} \cup U_{bc} \cup U_{ac}$$

2.2 Algorithms

This section describes different approaches of increasing complexity for taking on the pair classification problem. Active learning, which would require repeated user interactions for intermediate phases of the classification, is not considered as out of the scope of this work.

2.2.1 Classical

Naive approaches are based on equational rules for classifying pairs of records: the comparison vector is built to allow these rules to be applied. For example, several rules like *'matched if names are equal'* and *'matched if names differ by 1 letter and birth date is equal'* can be applied in succession after a priority for each of them is defined.

More sophisticated approaches build a comparison vector containing real numbers; each element of the vector representing a similarity between the values of the same fields in the different records.

The probabilistic approach applies the Expectation-Maximization algorithm on comparison vectors generated from each pair by comparing their corresponding fields. Each vector assumes a series of boolean values for field comparisons (0 for unmatched and 1 for matched) and a label $0 < g < 1$ corresponding to the probability of the pair being a positive match.

For instance, Jaro [19] applies the EM algorithm: in the E step classifications for the records are estimated from the current parameters of the distribution, while in the M step the log-likelihood of the data is maximized to give new values for the mixture distribution of matched and unmatched pairs.

Gu et al. [16] also uses the EM algorithm in the same way: the parameters are the conditional probabilities m_y and u_y to observe the agreement vector y given that the pair is a true match or a false match. Weigths are computed from the probabilities and used to update the status of pairs.

Jin et al. [20] explores also an approach of using attribute-specific distance definitions and predefined merging rules to map items on a multidimensional Euclidean space, while preserving similarity. Mapping on such a space is interesting for the subsequent usage of clustering algorithms that take advantage of the space properties to lower their computational complexity.

Transitive closure

In any case, classification of pairs alone does not solve the record linkage problem. A *transitive closure* step is always applied as the final operation in order to link chain of pairs into a single person: two matched pairs (a, b) and (b, c) may then become a single entity {a, b, c} even if the (a, c) pair has not been classified as matched or has not been classified at all.

Hernández and Stolfo [17] and Monge and Elkan [22] are examples of the popular usage of a union/find data structure to efficiently compute the transitive closure. Any pairwise matching algorithm can take advantage of this data structure.

The disjoint-set data structure is initialized with each element as a single subset; it then provides two operations:

- determine in which subset an element is.
- Join two subsets in a single one.

Even in a simple linked list implementation, these operations are respectively linear with the size of the subset (a list has to be traversed for each item) and constant (two linked lists are joined.)

The data structure is initialized with all the entities to reconcile, and then a union is performed between the components of each pair deemed as matched. A single pass over all entities then maps them to the subset they are in, which corresponds to a physical person.

Computational complexity reduction

The main computational problem in record linkage is that the size of the comparison space increases quadratically with the number of records considered. For all non-trivial problems, it is then unfeasible to consider *all* possible pairings between entities, an $O(N^2)$ operation.

Gu et al. [16] lists two alternatives for lowering the cardinality of the comparison space, named blocking and windowing.

As shown in figure 2.1, **blocking** reduces the pairs to the ones generated between records with an identical value on a particular blocking field, which is revealing of false matches but not necessarily of true matches. The A and B sets are partitioned into disjuncted blocks: for example, blocking on ZIP code results in only comparing people which live in the same city, a move that widely reduces the computational complexity of the classification, hopefully without removing matched pairs from the comparison space.

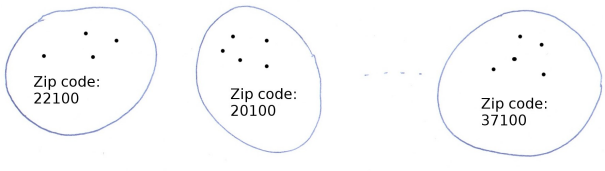

Figure 2.1: Blocking example

As shown in figure 2.2, **windowing** (also known as the *sorted neighborhood method*) prescribes the movement of a fixed-size window of W records over the list of sorted instances; only pairs contained in the window are considered. For example, sorting a list of people by name is going to exclude most of clearly different names from the comparison, such as *John* and *Thomas*; but including in

the set of pairs similar names such as *Gianni* and *Giovanni* for further analysis. Multiple passes of windowing on different sorting fields increase the comparison

Bianchi	Aldo
Bianchi	Alessandro
Bianchi	Alex
Bianchi	Federico
Bianchi	Luca
Bianchi	Matteo
Bianchi	Matteo
Bianchi	Mattia

Figure 2.2: Windowing example

space at a linear cost, which is preferrable to the quadratic cost of considering all possible pairings. In case of multiple passes, the union of the generated pair sets is considered.

In both blocking and windowing all non-considered pairs are deemed as unmatched pairs: the number of pairs increases linearly with the passes, which are however a bound number.

The application of either blocking or windowing trade-offs recall for performance reasons; they are both techniques statistically bound to found less matched pairs because the most distant records will be excluded by the comparison space. Nevertheless, they are able to reduce by 100x or 1000x the number of comparisons to perform (from quadratic with the number of records, to even linear in the windowing case).

Hernández and Stolfo [17] uses multiple windowing passes and a final step to compute the transitive closure between all matched pairs. The fields used as sorting keys vary from the principal key (last name) to street address. Experiments with window sizes from 2 to 60 confirm that multiple passes can achieve precisions in the range of 90% and that these values are aymptotic and do not increase when window size increases more.

Monge and Elkan [22] enhances multiple passes with different sorting fields by scanning the list with a priority queue keeping references to a few (4) last clusters detected.

2.2.2 Clustering

Clustering is an unsupervised approach to find matched pairs. The simplest clustering algorithm that can be applied to the problem is the agglomerative hierarchical one, where records (and not pairs) are recursively merged into clusters representing people, starting from the nearest pair of profiles and continuing at each iteration with the profile nearest to an existing cluster.

Figure 2.3 shows an example of profile clustering that aligns 8 profiles as 3 users.

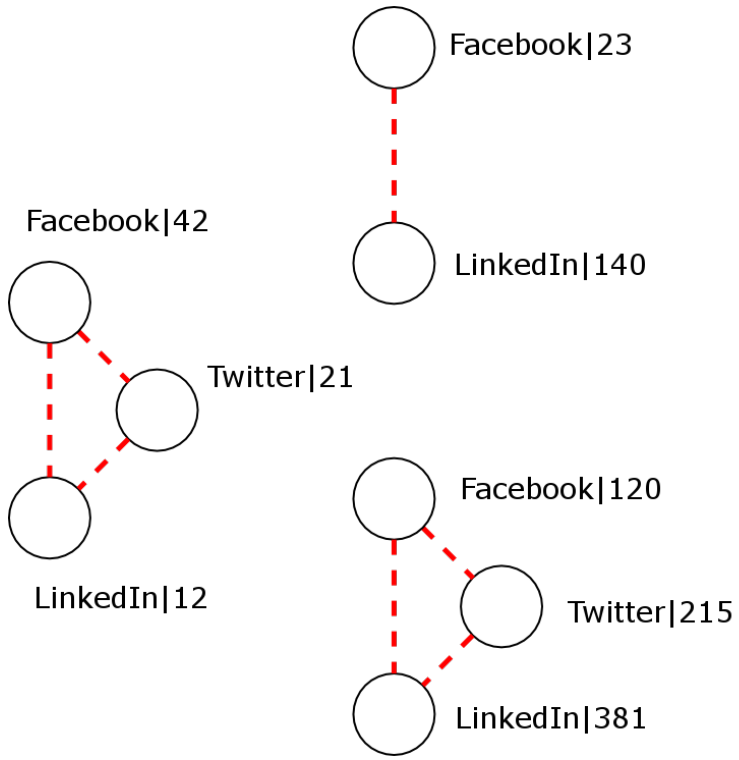

Figure 2.3: Clustering of profiles example

The single linkage variation can be computed in $O(N^2)$, which makes clustering difficult to use on large datasets. Windowing can improve the complexity by providing a list of similarities between entites sorted in descending order, that the agglomerative algorithm can read in sequence. In this semplification, every pair not in the list is then considered at infinite distance.

Naumann and Herschel [23] codifies this clustering approach followed by two variations of post-processing. Before applying transitive closure, post-processing

helps breaking up clusters that are unreasonably big and therefore cannot represent just a single entity.

Post-processing starts from a graph where entities are nodes and the matched pairs found via clustering are edges. Each edge is weighted with a single similarity measure where provided, or with a constant value in case the classifier does not provide a confidence value.

The first post-processing approach is partitioning based on **connected components**. This post-processing recursively removes the edge with lower similarity (or highest distance) until each connected component size is lower than a threshold (for example 3, which corresponds to 3 entities at maximum making up a single person.)

The second post-processing approach is partitioning based on **centers**. Edges are sorted in descending order of similarity and their list is traversed: every time a new node is encountered, is declared a center. All the remaining nodes connected to a new center are assigned to it and never considered again, and the list scanning goes on until all nodes have been considered, either as centers or as part of the cluster formed around a center.

Other approaches in clustering comprehend farthest-first clustering, where new users are created by targeting the farthest nodes from the last considered cluster. Goiser and Christen [15] use farthest-first to improve the performance of unsupervised (in this case clustering) learning on unblocked data, where K-means fails.

Even more sophisticated clustering approaches like sparse neighborhoods use a different distance or size thresholds for each cluster. Cohen and Richman [8] use a canopy method to generate the candidate pairs: this clustering method is more adaptive and conservative as a subset of the assigned points to a canopy can be reused and assigned to other ones (the farthest ones).

Verykios et al. [32] is still an example of unsupervised learning in that it uses clustering first to find out the labels of an otherwise unlabeled set of pairs, and then builds a decision tree which is broken down into rules.

2.2.3 Supervised learning

Supervised learning approaches start from a ground truth containing a mapping of a set of pairs from the same domain (census data, social networks or such) to

booleans defining them as matched or unmatched. The hope of supervised learning is to statistically learn a model that generalizes over the same domain and is able to classify new pairs.

Ground truth is not always available and that restricts the applicability of supervised learning; however, these approaches open up new possibilities for choosing the parameters of the model such as the importance of the distance between names with respect to the distance between birth dates or current locations (in general, between fields of the comparison vector).

Decision trees

Decision trees are one of the easiest to interpret model that can be learned from data. Given a training set containing for each pair:

- a list of distances between correspondent fields as feature values
- a classification as matched or unmatched

the C4.5 algorithm is able to learn thresholds for each distance and the order in which they should be considered in the decision rule. The end result is a tree where each branch is split basing on a feature value, as shown in the example of figure 2.4:

```
outlook = sunny
|   humidity <= 75: yes (2.0)
|   humidity > 75: no (3.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)

Number of Leaves  :     5

Size of the tree :      8
```

Figure 2.4: Decision Tree for weather prediction

The algorithm chooses the feature with the higher normalized information gain at each split, and performs pruning of complex branches to avoid overfitting and improve the legibility of the model.

In the realm of record linkage, Neiling [24] compares decision tree induction with classic record linkage on real data (250,000 German people) and conclude that it is

more accurate and robust than the latter. However, it is noted that decision trees do not allow to bound the false negative rate (matched pairs not classified as matched) nor to produce a list of dubious pairs for human examination; trees only produce a binary classification, and not a real score that could be compared with a threshold.

[1] uses clustering and decision trees to perform experiments in duplicate detection over synthetic data and over the set of Walmart products (200 millions of pairs), and results suggest that machine learning models outperform probabilistic (classic) approaches.

Support vector machines

Support vector machines estimates an hyperplane that separate the inputs in the feature space with the widest possible margin, or in the common case of non-linearly separable inputs minimizes the misclassified inputs. Linear support vector machines are considered here, such as the one of figure 2.5.

In the case of record linkage, inputs are pairs and the two classes to separate are still *matched* and *unmatched* in the same classification problem solved by decision trees.

SVMs have a greater complexity of training with respect to trees, but they also outperform state of the art trees in several empirical results.

```
Machine linear: showing attribute weights, not support vectors.

        0.8436 * (normalized) outlook=sunny
 +     -0.9535 * (normalized) outlook=overcast
 +      0.1099 * (normalized) outlook=rainy
 +      0.5276 * (normalized) temperature
 +      0.771  * (normalized) humidity
 +     -0.8901 * (normalized) windy
 -      0.8683
```

Figure 2.5: Support Vector Machine for weather prediction

Bilenko and Mooney [4] adopt SVM both for learning domain-specific string similarity measures (adapted to each field) and for formulating a predicate for pair classification by combining all measures according to learned weights. The claim is that this two-step approach outperforms non-adaptive ones as the similarites can be tuned to the data present in different domains instead of being statically chosen from a library of standard string distance metrics.

Christen [7] uses an unsupervised two-step approach where first training examples of high quality are selected as seeds and matched without a ground truth; second,

these pairs are used to train a SVM classifier. Newly classified pairs are iteratively added to the training set. The aim of this approach is to improve over other unsupervised algorithms, and not to compete with supervised ones where ground truth already exists.

2.3 Social network related work

Social network profiles matching and in general online profiles matching is not a new task. The literature contains several examples of work oriented to people and profile matching.

Carmagnola and Cena [5] build a framework for the interoperability of different systems (not public social networks) that ask to each other info on new users that have registered to retrieve more data for customization purposes. They use names, mails, and birth date as identifying fields and set model parameters by hand (the weights of matching field and the weight of how that field has univocity or multiple values per user).

Each field importance with respect to to other fields is learned from the data. A linear model is built with a threshold learned via validation on a 100-person dataset.

The domain of application is restricted as it consists of user-adaptive systems where performance is improved by real data and the user is incentivized to enter real data to get back appropriate recommendations or information.

Zafarani and Liu [35] study the identity elicitation problem: the search for a single user on different platforms. This is a username-based analysis, which involves pseudonyms and urls that contain these pseudonymes; no anagraphical data is analyzed for lack of availability. The average accuracy reported is of 66% across 12 online communities.

Vosecky et al. [33] use manual settings for the weights of different fields, but full profile vectors including information as name, home town, birthday, website or gender (but no connectivity information). The study is based on Facebook and the German-based network StudiVZ. The study reports an 83% success rate in identifying duplicate users as the evidence that the problem is solvable.

Carmagnola et al. [6] perform user elicitation via crawling: the input is a single user. The approach is based on public data available through crawling of HTML

pages and not on published APIs. Weights for fields similarities are chosen from a priori probabilities of the field values: for example, a match on gender is much less restricting than a match on names as the a priori probabilities of the *male* and *female* values are very high (both 0.5).

Iofciu et al. [18] use a combination of username similarity and tags similarity in a mixture model, modelling both explicit information (like anagraphic details) and implicit information like the tag used by a profile to categorize bookmarks on Delicious. The metric used for success is the rank at which the input user appears in elicited list, as in this context it would be very difficult to provide an exact match with this limited information.

Tang et al. [31] perform statistical analysis on names in Facebook's New York network, recognizing that first and last name are features that cannot be hidden by privacy on this particular platform. This observation is valuable and reflect a fundamental attribute of some social networks: in this case Facebook, but even more so for LinkedIn as in a professional network the user is expected to enter his real name.

The study is not focused on reconciliation but only to gender prediction and statistical analysis of names.

Raad et al. [28] is the most similar work in the state of the art, as it defines the same base problem of merging profiles from different platforms. However, this work defines a framework for evaluating unsupervised decision making algorithms, and the evaluation dataset is synthetic (having been randomly generated).

Bartunov et al. [3] define the concept of seed user and works on a single ego network: neighborhoods of the same person in different platforms. The study focuses on first neighborhood samples but could be in theory expanded to larger networks with the use of more anchor nodes.

Anchor nodes can even work as a de-anonymization tool as the graph signature of a node in one of the graphs with respect to anchor nodes can make it identifiable in other platforms. In this study, performance (mainly recall) is very influenced by the presence of at least some anchor nodes, due to the kind of similarities used.

To the best of our knowledge, no previous work in the literature provides an exaustive experimental study for the adoption of record linkage techniques in online social networks, comparing multiple classifiers and their variations over a real dataset, and attempting to automatically learn model parameters from the data in-

stead of requiring a human decision.

3 Implementation experience

This section describes the techniques evaluated by this work in the context of online social networks. First, the techniques under evaluation are listed along with the available record features for the classification problem. Then, the architecture and design of the application are presented.

3.1 Methods

post-processing	Clustering	Decision trees	Linear SVMs
No post-processing		X	X
Connected components breakup	X	X	X
Single profile per platform	X	X	X

Table 3.1: All variations experimentally evaluated

Different techniques and variations have been compared experimentally. Table 3.1 shows all the combinations that have been experimented with, and compared column-wise (post-processing variations for a given technique) and row-wise for what regards decision trees and SVMs (best technique given a post-processing strategy).

3.2 Distances

Several pair-to-pair distances from the literature have been defined on the available data, to be used as features for linking profiles, or classifying pairs:

- Levenshtein distance between real names.
- Jaro-Winkler distance between real names.
- TF/IDF distance between real names.
- Url distance between normalization of homepage fields.
- Levenshtein distance on other anagraphic fields such as: normalized birth date, location, nickname.

The **Levenshtein distance** is a simple edit distance for strings, that measures the amount of characters to remove or replace to transform a string into another.

The **Jaro-Winkler distance** is a more complex edit distance designed for the specific domain of record linkage and in particular of person names.

The **TF/IDF distance** is a classic information retrieval concept: in this case the terms of documents are actually the words contained in the string field where it is calculated. The term frequency is almost always 1 as there are very rare repetition in person names; the inverse document frequency is calculated over the available dataset, making rare names or surnames such as *Sironi* stand out with respect to more common ones such as *Rossi*, *Smith* or else.

The **Url distance** is calculated as the Levenshtein distance after normalization of Url values; for example, prefixes like *http://* and *www.* are removed as a form of stemming.

Birth dates are also normalized to the *yyyy-mm-dd* form, with missing segments like the year substituted by question marks.

For consistency of interpretation, all measures are expressed as distances (the lower the better), even if some of them are calculated as similarities. The conversion is made where necessary by elevating to the -1 power.

Note that (also in literature) these dissimilarity functions are often called distances, but only some of them (like Levenshtein's distance) are actually metrics. For example, the Jaro-Winkler distance does not satisfy the triangle inequality and is then not a metric distance in the mathematical sense.

Some additional features introduce domain-specific dissimilarities:

- Conflict: 1 if profiles are on the same platform, 0 otherwise.

- Common seed identity: 1 if the seeds are equal, 0 otherwise.

For what regards the *conflict* dissimilarity, it serves to exclude that two different profiles that reside on the same platform may belong to the same person. There are possible limit cases in where a real person has created multiple accounts on the same platform, but the datasets confirm that these are a few cases in the whole 30K profile set and may then be ignored for the sake of correctly classifying the majority of pairs.

The *seed* dissimilarity attempts to include a simple measurement for the presence or absence of a connection in the social graph. If two profiles point to the same seed, at least a common friend (the seed itself) exists between them; then this feature assumes the 1 value. We cannot say anything in case the seed identity is different, and the feature assumes the 0 value. This information is always freely available from the way the dataset is extracted from platforms (by using a relatively small set of authorizing user as seeds and crawling their neighborhood) and as such it does not add weight to the retrieval process, which is network-intensive.

Unknown (null) values are assigned for distances that cannot be calculated due to missing information (usually the absence of one of the two fields). The learning algorithms are supposed to be capable to deal with the missing value.

The datasets are stored as MongoDB BSON databases and as such conceptually in the JSON format. However, they can't be published due to personal details being present and to the Terms Of Service of the various platform from which they originate. Due to the particular nature of the features of the profile pairs, involving full names and nicknames, the anonymization of the datasets would prevent their usage.

3.2.1 Windowing

Pair generation is a common step for all the techniques under study, and takes place in the same way for both the training and validation sets, where the latter is present. Generating all the possible pairings between profiles is quadratic with the size of the dataset; the windowing approach from the literature generates instead a number of pairs linear with the size of the dataset and the fixed size of the window (20 profiles).

Windowing is preferrable to blocking due to the application domain: blocking

assumes there exist at least a field that uniquely identifies false match pairs when its values are different between the profiles (like a ZIP code or a nationality field). However, for most of the available data each field has many missing values or is expressed in non standard formats; for example *location* may assume the value *Milano* or *Milan area, Italy* for two profiles belonging to the same person. Thus blocking is not applicable in this domain due to the lack of a grouping attribute in the profiles.

Two windowing passes are executed on the profile set, by sorting records on the *name* and *website* field. Only pairs whose profiles are inside the same N=20 window in the sorting are considered, excluding of course the pairing of a profile with itself.

An advantage of consistently performing windowing is to make the prior probability of labels comparable: in the full pair set the probability of a pair to be a true match decreases quadratically with the size of the profile set. When windowing is present, the total number of pairs is linear and since only false matches are (hopefully) excluded, the prior probability of the true match class raised very much.

For example, in the larger evaluation dataset available, a single pair sampled from the full pair set has a 2.5E-6 probability of being a true match and as such an almost 1 probabiity of being a false match. When sampled from the windowing-based pair set, it has instead a 0.05 probability of being a true match.

Windowing has however to be kept consistent between training and usage of the classifier on a new dataset, to avoid skewing the prior probabilities the classifier has been trained on.

3.2.2 Classifiers: Clustering

Agglomerative hierarchical clustering was performed on the dataset as a form of unsupervised learning. The distance chosen was the Manhattan distance combining the Levenshtein distances over the *name* and *url* fields.

Windowing provides the clustering algorithm with a list of distances between points which is sorted in descending order, so that the algorithm has only to evaluate such a list instead of computing all the possible pairings by itself. All pairs not contained in the list are thus considered at infinite distance and never considered matched.

Each pair (that constitutes a linkage between points inside a cluster) consists of the identifiers $\{(platform_1, id_1), (platform_2, id_2)\}$ mapped to a floating point value D, where in this implementation $0 \leq D \leq 5 \cdot 2$.

Two post-processing techniques are experimented with in the clustering approach:

- As described in section 2.2.2, *connected components partitioning* breaks the clusters bigger than 3 profiles into smaller users (which would certainly contain at least a misclassified pair as there are only 3 different platforms in the dataset).
- *single profile per platform partitioning* breaks the cluster recursively by until each resulting user is not composed of multiple profiles belonging to a single platform.

The results in the absence of post-processing are not evaluated as this option is already ruled out by the state of the art as ineffective.

Single profile per platform can be generally applied only when it is known that there are no duplicates in the different source sets of profiles. This is true for the web-based platforms considered in this work, as they provide means for finding and retrieving access to existing accounts. The empirical data validates this assumption, as a proportion of less than 10 duplicated profiles on the same platform has been verified over a dataset of several tens of thousands profiles.

In the *single profile per platform* variation, each user formed by linking pairs judged as matched is processed independently; until there is a conflict in the mesh set of pairs generated by the user, an link between two pairs is iteratively removed.

For example, two Twitter profiles may be found in the same user, either because of a direct link or because of them being linked to a common third profile. In this case, the user is broken up until the two profiles are not connected anymore, not even indirectly.

To achieve the final result, the transitive closure is calculated so that all matched pairs inside a cluster are merged as a single user.

3.2.3 Classifiers: Decision trees

Decision trees are used as classifiers for directly assigning the windowed pairs to the matched or non matched class. Trees combine all the available features

(distances, conflict and seed identity) and learn the importance of each feature from the available training data.

The C4.5 algorithm builds and prunes a tree from a given training set consisting of feature values and *matched* or *unmatched* labels. All the Weka default options are set.

The confidence threshold for pruning has been held to the default 0.25, and it does not experimentally influence the accuracy of the result when varied from 0.1 to 0.75. Tuning this parameter too would require a separate test set for comparison with other methods, as its choice may overfit the validation set (which is used here only for gathering metrics and not for selecting any parameter or structure of the tree).

The decision tree classifier does not only consists of a tree: post-processing takes place, on both training and validation sets. As a control, the *no partitioning* variation and is evaluated, to verify the effectiveness of the *connected components partitioning* and *single profile per platform* strategies, as described in 3.2.2.

Summing up, the partitioning variations are:

- the already described single profile per platform partitioning.
- connected components partitioning: pairs are recursively removed until each user does not contain more than 3 profiles (the number of different platforms).
- no partitioning: used as a control group.

All the remaining valid pairs are merged to form users via the transitive closure.

3.2.4 Classifiers: Linear SVM

Support Vector Machines are general purpose classifiers and thus can be swapped in instead of a trees and leverage the same features. In fact, the post-processing steps available to SVMs to compute a transitive closure of pairs are the same as for trees.

The learning algorithm used is Sequential Minimal Optimization, with a polynomial kernel and a maximum esponent of 1 (which corresponds to a linear model). The C complexity constant is varied in the interval $[1 \times 10^{-2}; 1 \times 10^{2}]$, but does

not influence the accuracy of the result, and as such is not discussed further as a parameter. The other Weka defaults (normalization of all features and replacement of missing values) are maintained.

Due to the functional equivalence to trees, SVMs support the same post-processing variations:

- single profile per platform partitioning.
- connected components partitioning.
- no partitioning.

The windowing and the transitive closure computation steps are shared with the decision tree process, as described in section 3.2.3.

3.3 System model

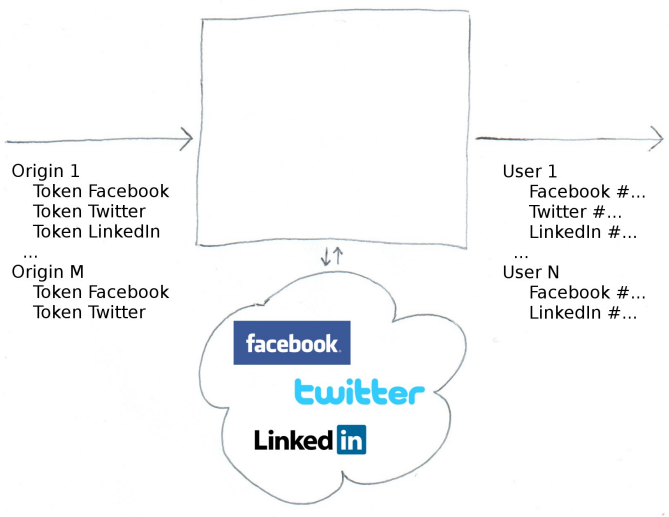

Figure 3.1: Black box model of the application

As shown in 3.1, the input of the system is a set of *seeds*. Each seed is a set of from 1 to P authentication tokens to access the API of an online social network on behalf of a user. P is the number of different platforms available; the access tokens of each seed are issued for hypothesis by the same person and are the only way for a 3rd party application to access data stored on the online platforms.

The output of the system is a set of N users. Each user represents a single physical person and contains from 1 to P profiles on the defined social networks. Each profile can be identified in the output by just the name of the platform and the platform-specific id; if other data is available, it is easy to link to the profile by these two informations.

Figure 3.2 shows the profile linkage data flow. The User Repository provides a set of profiles, and windowing is run over them to produce a set of possible pairs.

A distance value for each of the D_i features is calculated and attached to each pair. Then, a classifier divides the pairs in matched or unmatched, and a post-processing phase builds the transitive closure while trying to avoid to build users composed of an high number of profiles, which is not consistent with reality.

Figure 3.3 shows the data flow for the training of classifiers. A User Repository containing ground truth information extracts a training set composed of users, instead of single profiles. The profiles are then extracted and a windowing pass produces a large set of pairs, ignoring the ground truth and behaving as if it was not available.

A distance value is again calculated for each of the D_i features and the classifier training algorithm is fed the pairs with their associated distances, and the label for each. The output of the process is a classifier instance with all its parameters (e.g. weights for SVMs). The training process is not present for clustering, being an unsupervised approach.

The validation process in figure 3.4 is very similar to the main linkage process. However, pairs are produced by windowing in the training process. Moreover, there is an additional final step that compares the output of the process (the list of users) and compares it with ground truth.

Typical values for these parameters are:

- $P = 3$ e.g. Facebook, Twitter, Google+
- $M = 10$ as N will grow much quicker than M.
- $N = 10000$ or more.

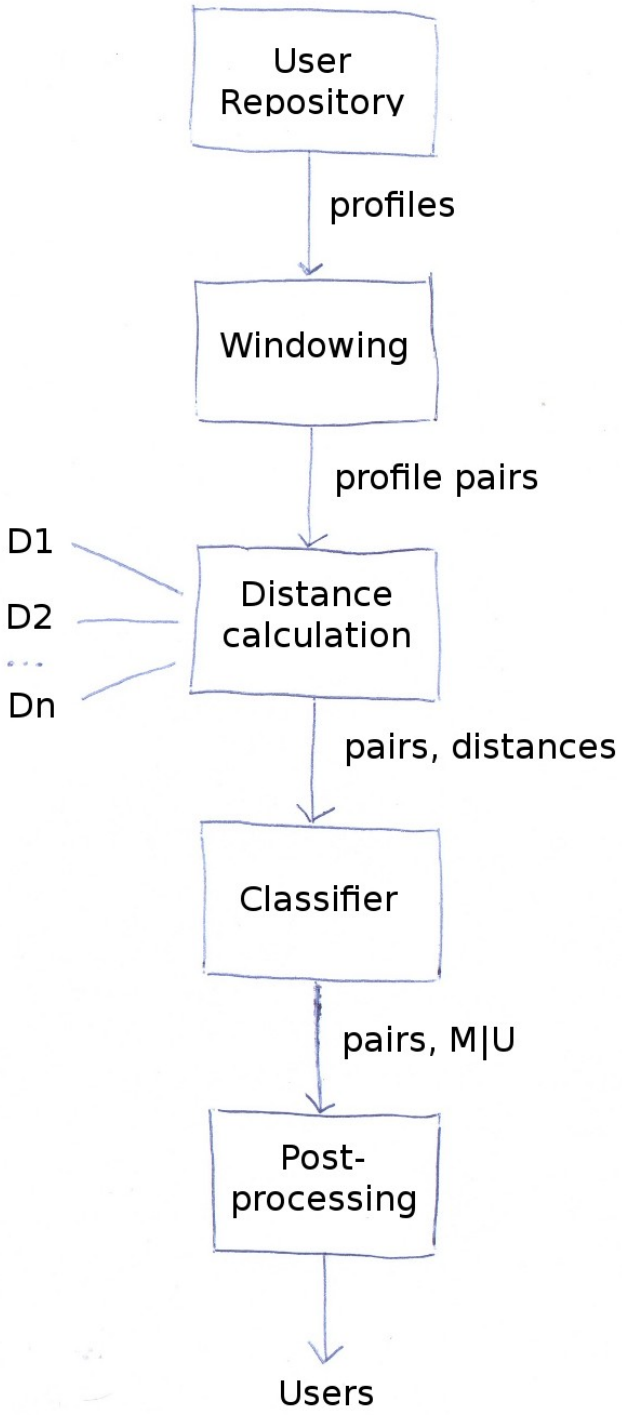

Figure 3.2: Profile linkage data flow

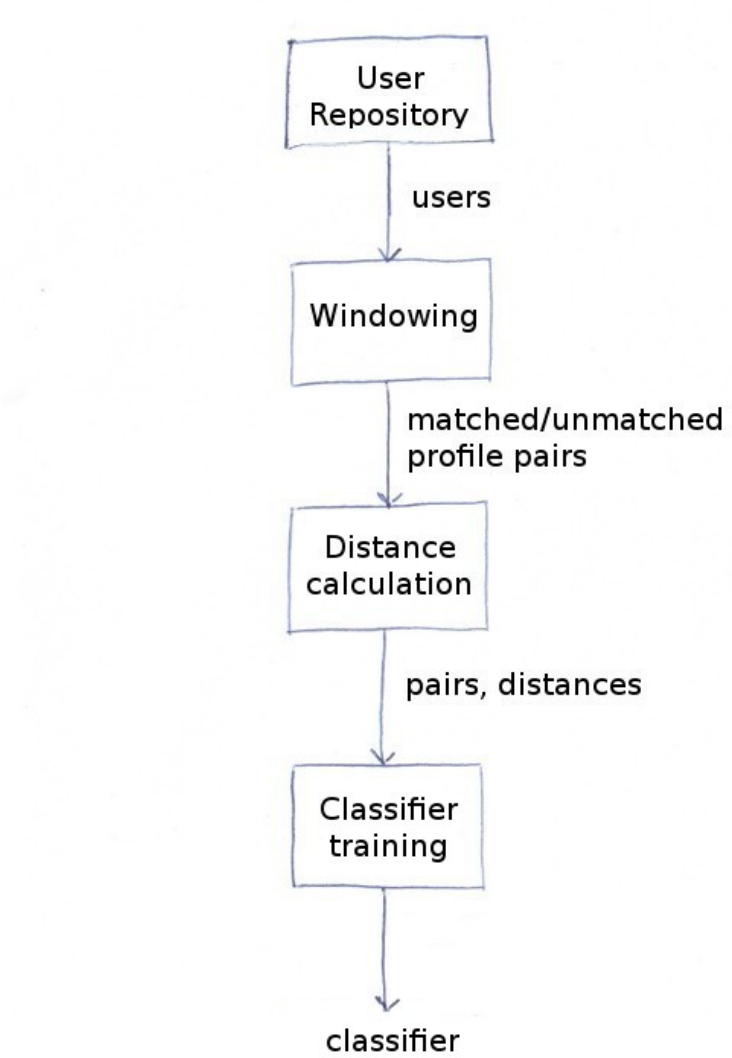

Figure 3.3: Training data flow

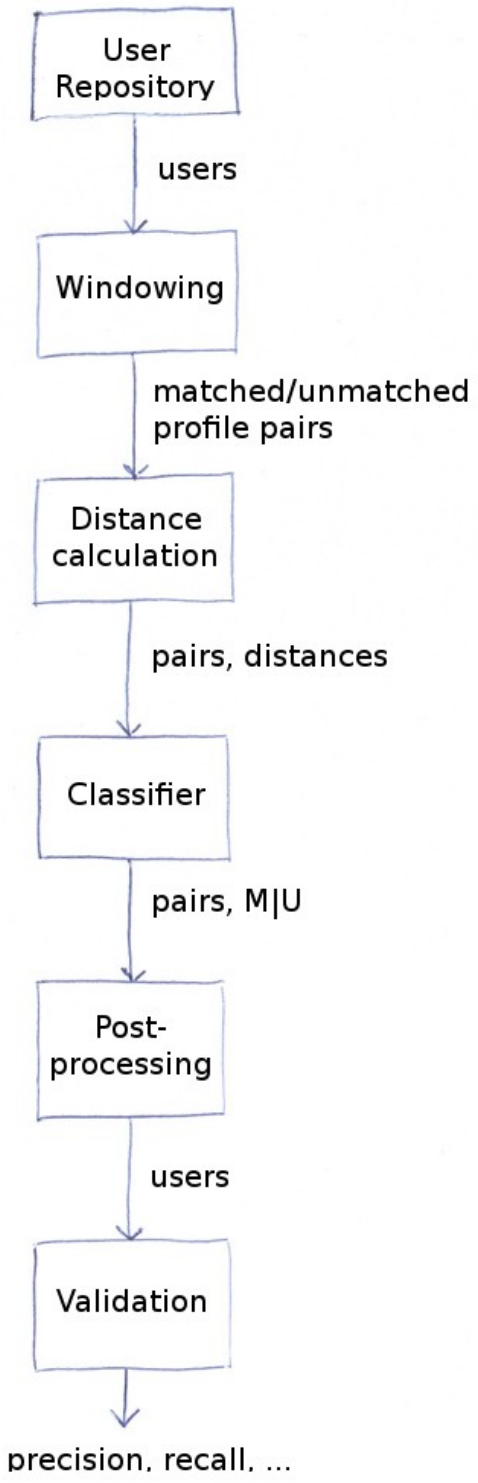

Figure 3.4: Validation data flow

3.4 Design

Each platform provides its own proprietary APIs containing a resource representing each profile. For example, Facebook provides a list of anagraphical fields depending on the level of privacy set by the profile's user:

```
{
    "id": "1588293761",
    "name": "Giorgio Sironi",
    "first_name": "Giorgio",
    "last_name": "Sironi",
    "username": "piccoloprincipe",
    "gender": "male",
    "locale": "en_US"
}
```

Twitter provides a monolithic set of fields for each profile:

```
{
    "id": 47998559,
    "favourites_count": 86,
    "profile_image_url": "http:\/\/a0.twimg.com\/profile_images
        \/427108002\/io_jamaica_q_normal.jpeg",
    "profile_background_tile": false,
    "profile_sidebar_fill_color": "CCF0FE",
    "verified": false,
    "location": "Como, Italy",
    "utc_offset": 3600,
    "name": "Giorgio Sironi",
    "lang": "en",
    "is_translator": false,
    "default_profile": false,
    "profile_background_color": "CCF0FE",
    "protected": false,
    "contributors_enabled": false,
    "time_zone": "Rome",
    "profile_background_image_url": "http:\/\/a0.twimg.com\/
        profile_background_images\/39231595\/twitter_bg4.jpg",
    "profile_link_color": "0084B4",
    "description": "Bachelor of Computer Engineering, web
        developer. Blogs on Invisible to the eye about [web]
        programming and engineering.",
    "geo_enabled": false,
    "listed_count": 128,
    "show_all_inline_media": false,
```

```
24      "notifications": null,
25      "id_str": "47998559",
26      "statuses_count": 5195,
27      "profile_image_url_https": "https:\/\/si0.twimg.com\/
            profile_images\/427108002\/io_jamaica_q_normal.jpeg",
28      "following": null,
29      "profile_use_background_image": true,
30      "screen_name": "giorgiosironi",
31      "profile_background_image_url_https": "https:\/\/si0.twimg.com
            \/profile_background_images\/39231595\/twitter_bg4.jpg",
32      "followers_count": 1228,
33      "profile_text_color": "333333",
34      "follow_request_sent": null,
35      "friends_count": 417,
36      "url": "http:\/\/giorgiosironi.blogspot.com",
37      "created_at": "Wed Jun 17 15:35:30 +0000 2009",
38      "default_profile_image": false,
39      "profile_sidebar_border_color": "C0DEED"
40  }
```

LinkedIn provides an highly customizable output, still depending on the profile's privacy settings and on the requested fields:

```
1   {
2       "values": [{
3           "headline": "Developer Advocate at LinkedIn",
4           "id": "4Lpna3NZkr",
5           "lastName": "Jones",
6           "pictureUrl": "http://media.linkedin.com/mpr/mprx/0
                _8LXTPUg",
7           "_key": "~",
8           "firstName": "Kirsten"
9       }],
10      "_total": 1
11  }
```

In addition, all APIs provide other resources, such as posts and groups, that can be linked to each profile by using its ids.

However, a minimum common denominator has to be extracted from these APIs as attributes of a profile are only relevant for profile linkage if they are present on multiple platforms. Many fields have to be discarded instead because of being platform-specific (the creation date or customization settings).

There are also several issues in extending the profile attributes:

- Privacy settings may cut out the current access token from reading some of the attributes.
- Completeness is not guaranteed: many users leave explicit profile fields blank.
- When the desired data are available from a different resource from the profile's URL, additional network calls for each user linearly increase the retrieval time.

The integration of profile data from the chosen platforms (Facebook, Twitter, LinkedIn) is consistently carried out with OAuth, the open standard for 3rd party authorization.

In the OAuth framework, users allow access to their data on a service provider (a platform like Facebook) without sharing their access credentials with the 3rd party. Instead, a random token is generated in the process and authorized for usage by a dialogue between the user and the service provider. The 3rd party only gets to know the token and can access the user's data only within the limits set by the user on the token itself.

The system metaphor corresponds to a user issuing a valet key for the 3rd party, in the example of figure 3.5 being an Android application. The valet key has limited permissions and duration with respect to the full key.

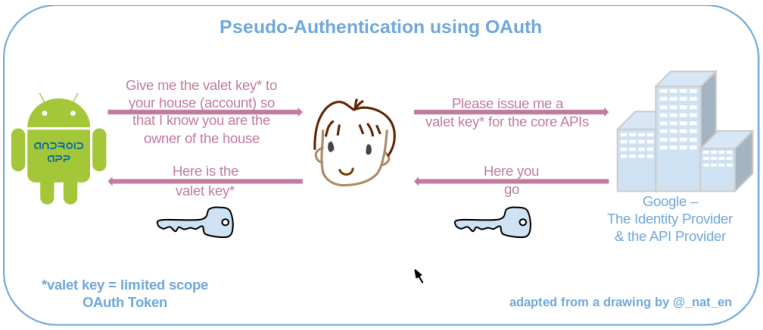

Figure 3.5: OAuth metaphor

Tokens begin their lifecycle in the 3rd party, that generates them either via an SDK or by asking directly the service provider for one. Users are directed on a trusted page on the service provider's website where they can authorize this token.

A few (dozen) seeds are used by the application as a point of entry: their own profiles are crawled along with their more massive circle of connections, following or friends. A few tokens can easily allow access to some hundreds more profiles.

4 Evaluation

The evaluation is described as follows. First, the gathered datasets are presented, and the ground truth manual generation method is documented. Second, the evaluation methodology for the classifiers and their comparison is defined. Third, the experimental results for the various models are presented and discussed.

4.1 Datasets

Each seed user allows the application to act on his behalf with the available platforms, resulting in the profiles of him and his friends being available for extraction. The tokens of each seed are correlated and stored in this form:

```
{
    "_id": 20,
    "facebook": "AAABjAfwd...azfPLQZD",
    "twitter_token": "16182078...eOC3UVK8",
    "twitter_secret": "D797IKOh...5nDCBeMM",
    "linkedin_token": "f4761832...229fe8d0",
    "linkedin_secret": "3d1f80c8...8c4bbc77"
}
```

The circle of users surrounding the seeds is gathered by stopping at the directly connected nodes of the social graph (distance 1 from the seed).

This work has been part of the Cubrik project described in Fraternali et al. [12], and the Cubrik data model has been extended with:

- a profile-specific name considered for each user, to allow different declina-

tion, abbreviations, and nicknames.

- a seed identity field is added to each profile to consider the provenance of the data.
- Only populated fields are included into the model, as a simplification over the full one.

After the retrieval process, the input data for the reconciliation process is a list of separate users, each with a single profile. Before reconciliation a single person may appear multiple times in the list, one for each of the platforms where he has registered.

```
{
    "_id": ObjectId("4ff706fee4b03be499e0f365"),
    "membership": [{
        "platform": "facebook",
        "platform_profile": {
            "id": "1588293761",
            "profile_url": "http://www.facebook.com/
                piccoloprincipe",
            "nickname": "piccoloprincipe",
            "name": "Giorgio Sironi",
            "origin": 5,
            "date_of_birth": "1988-12-31",
            "attributes": {
                "location": "Como, Italy",
                "homepage": "http://giorgiosironi.blogspot.com"
            }
        }
    }]
    "name": "Giorgio Sironi"
},
{
    "_id": ObjectId("4ff6faeae4b0175013acce40"),
    "membership": [{
        "platform": "linkedin",
        "platform_profile": {
            "id": "eQIMxUpSV_",
            "profile_url": "http://www.linkedin.com/in/
                giorgiosironi",
            "nickname": "eQIMxUpSV_",
            "name": "Giorgio Sironi",
            "date_of_birth": "1988-12-31",
```

```
            "attributes": {
                "location": "Milan Area, Italy",
                "homepage": "http://giorgiosironi.blogspot.com"
            },
            "origin": 0
        }
    }]
},
{
    "_id": ObjectId("4ff711a9e4b027c7eae5e19b"),
    "membership": [{
        "platform": "twitter",
        "platform_profile": {
            "id": "47998559",
            "profile_url": "https://twitter.com/#!/giorgiosironi",
            "nickname": "giorgiosironi",
            "name": "Giorgio Sironi",
            "date_of_birth": null,
            "origin": 7,
            "attributes": {
                "location": "Como, Italy",
                "homepage": "http://giorgiosironi.blogspot.com"
            }
        }
    }]
}
```

Only the common subset of fields present on at least two platforms is present in this data model. Only profile fields are extracted for performance reasons; due to privacy and completeness issues described in 3.4, it is not proficient to retrieve additional data from the platform.

After reconciliation, the goal is to merge multiple profiles from the same person into a single entity (for all the available users that have more than one profile in the list; there is no elicitation of profiles that are not in the retrieved list.)

```
{
    "_id": ObjectId("4ffa7dfee4b0c2f8946c4095"),
    "membership": [{
        "platform": "linkedin",
        "platform_profile": {
            "id": "eQIMxUpSV_",
            "profile_url": "http://www.linkedin.com/in/
                giorgiosironi",
```

```
8               "nickname": "eQIMxUpSV_",
9               "name": "Giorgio Sironi",
10              "origin": 0,
11              "date_of_birth": "1988-12-31",
12              "attributes": {
13                  "location": "Milan Area, Italy",
14                  "homepage": "http://giorgiosironi.blogspot.com"
15              }
16          }
17      }, {
18          "platform": "facebook",
19          "platform_profile": {
20              "id": "1588293761",
21              "profile_url": "http://www.facebook.com/
                    piccoloprincipe",
22              "nickname": "piccoloprincipe",
23              "name": "Giorgio Sironi",
24              "origin": 5,
25              "date_of_birth": "1988-12-31",
26              "attributes": {
27                  "location": "Como, Italy",
28                  "homepage": "http://giorgiosironi.blogspot.com"
29              }
30          }
31      }, {
32          "platform": "twitter",
33          "platform_profile": {
34              "id": "47998559",
35              "profile_url": "https://twitter.com/#!/giorgiosironi",
36              "nickname": "giorgiosironi",
37              "name": "Giorgio Sironi",
38              "origin": 7,
39              "attributes": {
40                  "location": "Como, Italy",
41                  "homepage": "http://giorgiosironi.blogspot.com"
42              }
43          }
44      }],
45  }
```

A *small* dataset was gathered at the start of this work to gain a feeling for real data; a much larger dataset has been gathered for evaluation once the application was running. This larger dataset is named *expertfinding* because the seed users are

Name	small	expertfinding
Seeds	7	60
Profiles	3512	30614
Matched pairs	420	2100
Users	3092	28514
Users with multiple profiles	344	1920

Table 4.1: Gathered datasets

Dataset	small	expertfinding
Real name	100%	100%
Nickname	91.0%	98.8%
Location	75.2%	41.6%
Homepage	54.1%	28.7%
Birth date	45.0%	6.95%

Table 4.2: Completeness of profiles

borrowed from a different application evaluating expert finding strategies on online social networks.

The relevant measurements for these datasets are shown in table 4.1.

Table 4.2 shows the percentage of the profiles that contain a value for a particular field (whether that value is consistent with the field or not; for example nicknames can be inserted in the name field.)

As a common practice in the related work, an *instance* is defined as a pair of profiles, and it is then classified as belonging to the *true match* class or to the *false match* class. Features that can be used by classifiers to decide between the two labels are calculated over the pair, using information contained in both the profiles.

4.2 Ground truth

The ground truth for each dataset was built by hand, by duplicating the crawled database and performing two successive manual steps:

- windowing over real names ordered alphabetically, addressed to the neighborhood of each user.
- Special cases search on all non proper nouns (pseudonymes, nicknames, inverted last/first name) addressed to all the rest of the dataset.

All the evaluations of pairs deriving from the windowing step or the special case search are performed by a human. Fortunately, the parallelization of this process is trivial - simply dividing the windowing phase into different initial letters (matching names from A to Z) is enough to scale the generation process to 26 operators (although the distribution of the users over initial letters is not flat - further divisions like AA to AM and AN to AZ may be needed.)

The windowing step is necessary because the full comparison space is still too large for a human to handle: with a dataset in the order of thousands of users, millions of comparisons will be needed for a full evaluation of the space. The user interface for windowing is shown in figure 4.1.

To counter the minor recall provided by the windowing step, the special case search attempts to determine the real full name of every profile which happens to be named with a nickname or in any way different from the ordinary *FirstName LastName* scheme. For example, the Twitter user *@42John* may be determined to be *John Smith* from its profile. Note that matched pairs containing @42John may not appear in the windowing step, because *42John* and *John Smith* will be far from each other in the alphabetical order of profiles.

Therefore, each special case is looked up to determine a name and searched in the whole database for a comparison, as shown in figure 4.2. Usually, the first or last name search if enough to bring up profiles belonging to the same person which happen to contain a real name instead of a pseudonym. It it still not possible to link matched profiles which both use pseudonyms instead of real names.

For the purpose of speeding up the human linkage process, a web-based user interface has been built containing links to the real profiles for checking the identify of a profile with additional information, and facilities for merging profiles into a single recognized user.

While the user interface provides just names and seeds in its own screen, it is often possible to deduce additional information from the linked version of the profiles; online profiles contain photographs, interests and text that can solve the problematic cases.

Figure 4.1: User interface for the manual production of ground truth

Figure 4.2: Special case search for surname 'Sironi'

4.3 Metrics

The evaluation of the quality of a model is based on the information retrieval *precision* and *recall* metrics (as described for example in Manning et al. [21]) calculated over the set of pairs; instead of *retrieved* and *relevant* sets, this version of the metrics considers the classified as matched set, and ground truth matched sets.

In this framework:

- False Positives are defined as pairs of profiles that do not belong to the same person, but were classified as matched.
- False Negatives are defined as pairs of profiles that do belong to the same person, but were classified as unmatched.
- True Positives as matched pairs correctly classified.
- True Negatives as unmatched pairs correctly classified.

These categories are visualized as Venn diagrams in figure 4.3.

Precision and recall can then be defined as:

$$P = TP/(TP+FP)$$

$$R = TP/(TP+FN)$$

where precision is a measure of how many of the pairs classified as matched were actually True Positives. Conversely, recall is a measure of how many matched pairs were correctly classified as matched and not left out.

In scenarios where a single metric has to be obtained, the F-measure (or another combination of precision and recall) can be calculated starting from P and R values:

$$F = \frac{2 \cdot P \cdot R}{P+R}$$

The two precision and recall metrics are calculated over a portion of the dataset selected according to the learning algorithm under evaluation. The two datasets are always used separately as their ground truth has been generated accordingly.

In the case of unsupervised techniques (clustering), it is possible to pass the whole dataset (without any ground truth information) to the learning algorithm and eval-

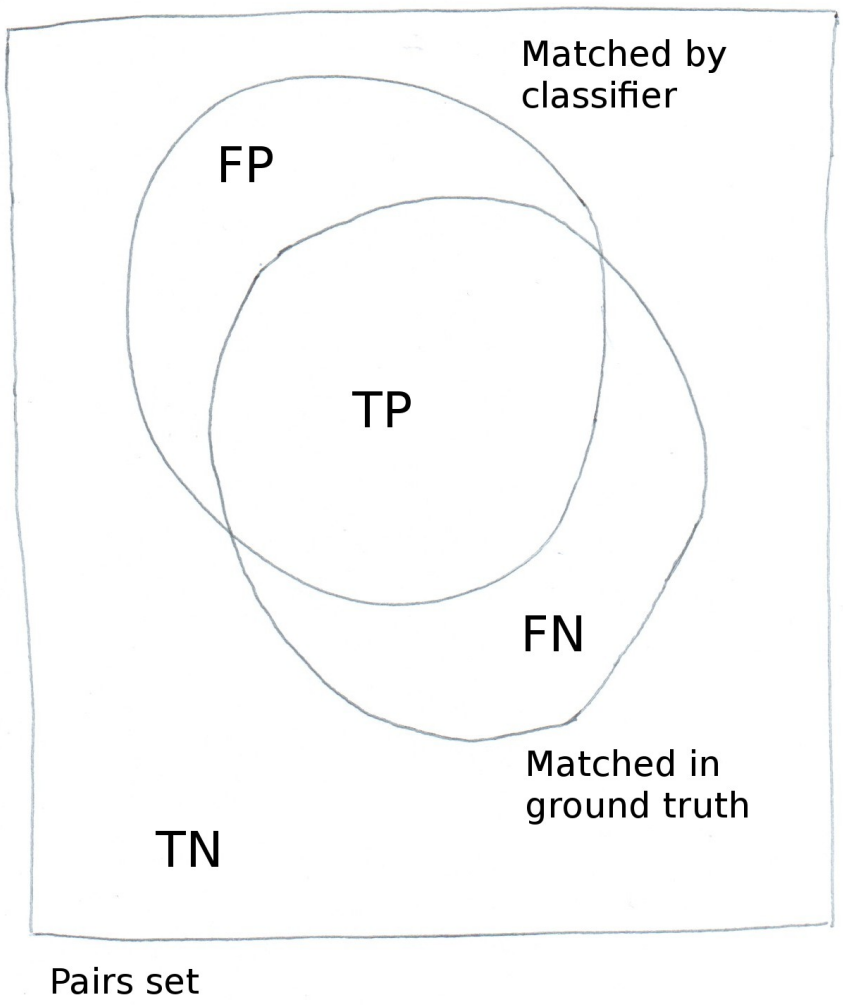

Figure 4.3: Visualization of false positives and negatives

uate the final result separately. Thus clustering has been performed over the whole datasets.

In the case of supervised approaches, it is known that evaluating a model on the same data used for training it produces a danger of overestimation of the model's predictive power. 5-fold cross-validation is thus employed to produce a realistic estimate of the models accuracy, by averaging the 5 values of precision and recall for each round of cross-validation.

This implementation of cross-validation is stratified to guarantee that the percentage of users with more than one profile is constant in each fold. The selection in each stratum is random but predictable, as reconciled users can be consistently ordered with their record UUID (randomly generated on insertion).

```
1  getFold(int index, int folds) {
2      return
3          sample(usersWithSingleProfile(), index, folds)
4              U
5          sample(usersWithMultipleProfiles(), index, folds);
6  }
7
8  sample(collection, int foldIndex, int folds) {
9      int size = collection.count();
10     int foldLength = size / folds;
11     int start = (index - 1) * foldLength;
12
13     contents = view.find().sort(byId()).limit(foldLength).skip(
           start);
14
15     set = {};
16     foreach (contents as fullUser) {
17         set = set U reconstituteSeparateUsersFromReconciled(
               fullUser);
18     }
19     return set;
20 }
```

In addition to the evaluation of single models, it is interesting also to perform a comparison between different versions of the same model where the post-processing policy vary between the no post-processing control strategy, the breakup of large connected components and the *single profile per platform* strategy.

However, performing statistical significance tests with the standard $\alpha = 5\%$ on all

the combinations of variations (such as C4.5 trees and SVM with or without post-processing) results in the problem of multiple comparisons Salzberg [29]. The significance level α of the tests corresponds to the probability of committing a type I error on that test; given that we perform N multiple tests in parallel, the probability of committing at least an error α_T is:

$$\alpha_T = 1 - (1 - \alpha)^N$$

Thus a significance level of 5% repeated over 10 tests that compare classifiers corresponds to an overall significance level of 40%, which is much higher than the original.

For this reason, a more stringent criterion is used for the tests: Bonferroni's correction.

$$\alpha = \alpha_T / N$$

where N is the number of classifier comparisons. The α value may be relative either to a one-tailed or two-tailed test.

The significance test suggested by Dietterich [10] is McNemar's test, which is similar and in some cases equivalent to the binomial sign test. When a ground truth is given:

$$G(i) = \begin{cases} 1 & \text{if } i \in M \\ 0 & \text{if } i \in U \end{cases}$$

A similar function can be defined for a classifier A:

$$A(i) = \begin{cases} 1 & \text{if i is classified as matched} \\ 0 & \text{if i is classified as unmatched} \end{cases}$$

To compare two classifiers A and B, a test statistic is calculated starting from the parameters:

$$a = |i | G(i) = A(i), G(i) \neq B(i)|$$

$$b = |i | G(i) \neq A(i), G(i) = B(i)|$$

Plainly speaking, a is the number of instances the A classifies correctly as matched or unmatched and B classifies incorrectly. Conversely, b is the number of instances that B classifies correctly as matched or unmatched, but A classifies incorrectly. By design, the instances where the two classifiers agree are ignored by the test.

Under the null hypothesis that the classifiers are equivalent, a and b should have roughly equal values. The binomial distribution can be used in an exact test to calculate the probability that a and b differ by *at* least the experimental amount:

$$X \sim Bi(0.5, a+b)$$

$$p = P(b-a \geq observedvalue | p(success) = 0.5)$$

that is, the probability of committing a Type I error by rejecting the hypothesis that the classifiers are identical.

Assuming $a < b$ with no loss of generality, and developing the expression of p:

$$p = \sum_{k=b}^{a+b} \frac{(a+b)!}{k!(a+b-k)!} \cdot 0.5^{a+b}$$

p is the value to compare with the significance α. The null hypothesis will be rejected only if $p < \alpha$.

4.4 Results

4.4.1 Accuracy

The following tables present the results for the *small* and *expertfinding* dataset, in the form of a precision and recall figure for each algorithm and variation. Only the expertfinding dataset will be considered in further discussion as the small dataset has been used for exploration and tuning of the distance measures, and as such is not disconnected from the training of the various techniques.

The post-processing variations are listed in table 4.3 and table 4.4 with the following labels as the table columns:

- *null*: no post processing.
- *connected*: breakup of connected components with size greater than 3.
- *single*: breakup of users until a single profile per platform remains.

	Precision Recall		
technique	null	connected	single
clustering	-	0.859 0.921(t=6)	0.954 0.929 (t=6)
decision trees	0.965 0.915	0.961 0.915	0.980 0.909
SVM	0.980 0.898	0.980 0.898	0.987 0.892

Table 4.3: Small dataset results

	Precision Recall		
technique	null	connected	single
clustering	-	0.673 0.995 (t=5)	0.866 0.877 (t=5)
decision trees	0.782 0.933	0.884 0.931	0.905 0.928
SVM	0.892 0.903	0.902 0.901	0.916 0.897

Table 4.4: Expertfinding dataset results

4.4.2 Significance

There are 7 possibile significance tests that can be performed on post-processing variations:

- 1 comparison of the two clustering variations
- 3 comparisons of the three trees variations (in pairs)
- 3 comparisons of the three SVM variations (in pairs)

Furthermore, there are 3 possible comparisons between different techniques:

- trees and SVMs with no post-processing
- trees and SVMs with connected components post-processing
- trees and SVMs with single profile per platform post-processing

for a total of 10 significance tests.

The critical value for McNemar's test is:

$$\alpha = \frac{0.05}{10 \cdot 2} = 2.5 \times 10^{-3}$$

because of the 10 tests to perform on techniques variations, and of them being two-tailed tests. Two-tailed tests are more robust for comparison of these classifiers

as often different variations increase recall while decreasing precision (and viceversa); the alternative hypothesis is then just that the classifiers are significantly different, and the winner of the comparison should be decided by the test itself.

Table 4.5 shows McNemar's statistical significance test shows that the *single* variation has a better accuracy when used after clustering.

variation A	variation B	A beats B	B beats A	p-value	significant
connected	single	80	894	$< 1 \times 10^{-6}$	yes

Table 4.5: Significance testing for clustering post-processing variations

Table 4.6 shows McNemar's test executed over the results of different variations of the decision tree classifier.

variation A	variation B	A beats B	B beats A	p-value	significant
null	connected	0	6	1.56×10^{-2}	no
null	single	4	24	9.00×10^{-5}	yes
connected	single	4	18	2.17×10^{-3}	yes

Table 4.6: Significance testing for trees post-processing variations

Table 4.7 shows McNemar's test executed over the results of different variations of the linear SVM classifier.

variation A	variation B	A beats B	B beats A	p-value	significant
null	connected	2	10	1.93×10^{-2}	no
null	single	9	33	1.36×10^{-4}	yes
connected	single	8	24	3.50×10^{-3}	no

Table 4.7: Significance testing for linear SVMs post-processing variations

Table 4.7 shows McNemar's test executed between the tree and linear SVM classifiers, each instantiated with the same post-processing policy to keep all other factors equal in the comparison.

Variation	Tree beats SVM	SVM beats tree	p-value	significant
null	86	46	3.16×10^{-4}	yes
connected	84	45	3.78×10^{-4}	yes
single	75	41	1.02×10^{-3}	yes

Table 4.8: Significance testing for trees and SVMs

4.5 Discussion

4.5.1 Clustering

Clustering shows adequate results, but there are several issues with this approach that makes it difficult to rely on.

The first issue is scalability: clustering algorithms are at least $O(N^2)$ if not $O(N^3)$ in some cases, depending on the type of linkage chosen. Computational complexity can be improved via windowing or blocking if we define all non generated pairs has having an infinite (or simply very large) distance: in that case the algorithm limits itself to consider the distance over the pairs set while searching for the next point to add to a cluster.

The second issue is the arbitrariness of many parameters, due to the unsupervised approach. Parameters have to be chosen for:

- the maximum distance between two profiles.
- The unknown distance value to use when one of the profiles in the pair has a null value. Both corner cases are mapped to 5 in this implementation.
- The weights for combining different distances, which are defined on an arbitrary scale.
- The threshold to reach as the distance between the next two profiles in order to stop clustering.

Different threshold levels of the clustering (0 to 9) influence the trade-off between precision and recall. In this implementation, the sweet spot seems to be around the 6 and 7 thresholds; this empirical threshold means that to correctly classify an high percentage of pairs at least one of the distances may assume a maximum value in a matched pair, while the other is a perfect or almost perfect match showing 1 or 2 edit distance.

Given a choice of threshold for clustering, there is a significant advantage in using the *single* variation for post-processing with respect to the *connected* one. Clustering is in fact mostly used in the literature in the context of duplicate detection in a single list of profiles, where the *single* variation cannot be applied at all.

4.5.2 Decision trees

Decision trees reveal a strong sensibility to the presence of post-processing and to its implementation:

- no partitioning results in the worse precision, due to the agglomeration of an high number of profiles into the same user and consequently due to the creation of too many artificial pairs in the transitive closure step.
- Standard cluster-size partitioning (the same as in clustering approach) results in 88% precision.
- The domain-specific conflict-based partitioning results in 90% precision.

Recall values are almost constant throughout the variations.

McNemar's test results declare that there is a statistically significant difference between the *single* variation and either the *null* or *connected* ones: the *single* variation achieves an higher precision. This results demonstrates empirically the need for post-processing and that choosing the best post-processing policy matters. It is unclear from these results whether the *connected* components partitioning is more effective as the *null* one or not (the null hypothesis stands.)

Boosting

Boosting is the linear combination of multiple high-bias decision trees for classification. This technique was investigated because of the potential for a weighted combination of trees to produce a confidence value as their output, instead of a single *matched/unmatched* label.

This further analysis was based on AdaBoost as implemented in Freund and Schapire [14] and ignored bagging of trees (which fixes weight to +1 or -1) because of the notion diffused by Quinlan [26] that boosting almost always outperform bagging.

Boosting proved not useful with C4.5 the trees as they are already too accurate to benefit: all the trees iteratively trained after the first are weighted with a coefficient near to 0. By augmenting the bias of trees different results may be reached.

An examples of weights for a list of trees obtained by boosting is:

1. 3.8547516858069897
2. 219.751911405549351E-5
3. 219.751911405549351E-5

and it is clear that only the first tree influences the final result.

4.5.3 Linear support vector machines

Support Vector Machines do not show much sensibility to the kind of post-processing chosen in terms of average precision or recall (coming from 5-fold cross-validation). However, McNemar's test show a significant difference between the *null* variation (absence of post-processing) and the *single* post-processing policy.

4.5.4 Comparison of different classifiers

Clustering is excluded from the comparison with other kinds of classifiers due to its lack of leveraging of all features of record instances.

McNemar's test comparing decision tree induction to support vector machines show a significant difference in the results of the two classifiers, signaling that trees always outperform linear support vector machines, for every given post-processing policy.

Trees achieve this result by having an higher recall in all cases, and approaching the SVM's precision when the best post-processing policy is available.

The pairs classified differently by trees and SVM are a larger set than in the case of post-processing variations for the same algorithm. This suggests that:

- the choice of classifier has a larger impact on the end result than the choice of post-processing policy.
- different kinds of problematic pairs are identified as matched by trees and SVMs.

Classifier	Training time	Validation time
C4.5 trees	90-100s	5s
SVM	10s	5s

Table 4.9: Running times for a single fold of the expert finding dataset

If we compare training times as shown in table 4.9, we can see that trees training lasts as much as 9-10 times the SVMs one. This suggests there exists a trade-off between time efficiency and accuracy in this context: trees can be used to maximize recall and accuracy, while SVM have slightly inferior results but a training time an order of magnitude inferior to trees. Moreover, execution times of a already built model are roughly comparable for the two methods. Therefore, execution time should not be used as a parameter to decide between the two classifiers.

These execution times have been gathered on an Intel Pentium E5300 (2M Cache, 2.60 GHz, 800 MHz FSB) and all data (distances included) is preloaded in memory or precomputed so that the training algorithms are CPU-bound. However, they have only been reported for a relative comparison of classifier performance and not as an absolute benchmark.

4.6 Technical issues

4.6.1 Optimization

Windowing

As explained in the background section, windowing is strongly necessary to reduce the number of pairs to evaluate as matched or unmatched. This kind of time optimization has been already extensively treated in 2.2.1 and 3.2.1.

Data structures

Due to the high amount of computation involved but relatively moderate amount of data to keep in memory, it is beneficial to optimize the data structures used in pre- and post-processing and in the implementation of algorithms for time with respect to space.

For example, all distances for each pair to be classified are precomputed and thus

calculated only once for each pair. Pairs and users are always stored in hash sets or hash tables with the goal of constant lookup time. The hash functions for them are defined as follows:

- User objects are hashed as the Java String class hash of their alphanumeric id, randomly generated by the underlying database.
- Pairs are hashed as the sum of the hashcodes of the $platform|id$ representations of the profiles they are composed of (e.g. $facebook|15162342$). This is a simple way to guarantee an equal hash value even for pairs whose contained profiles are inverted.

Data structures

The well-known union/find data structure is used in various post-processing variations to compute the connected components of the profiles graph. An instance of DisjointSet is created for each node in the graph (or in a cluster) and the union operation is performed for each matched pair. The union operation consists of setting one of the profiles as the successor of the other, establishing a new conceptual linked list.

The result is a series of inverted trees, one for each connected component: by putting the head of the tree into a map it is immediate to generate the number of different elements, which corresponds to the number of components.

In the example of figure 4.4, 5 profiles have been connected for a total of 4 union operation:

- A-B
- C-B
- D-C
- B-E

Each profile in the data structure can calculate the same head of the list E, and use it for disambiguating whether it has been (even indirectly) connected to one of the other given profiles.

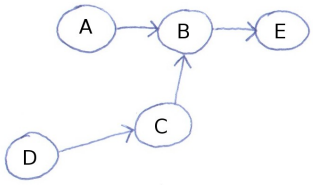

Figure 4.4: Union/find data structure for profiles

Caching

The UserSet object, representing a training or validation set, is one of the base data structures of the application. As such, it is a bottleneck for all operations that build on users, profiles and pairs.

The UserSet however is rarely modified once data have been retrieved from the database and User objects have been put in it. As such, it caches the following objects:

- The list of all pairs of the users in the set (thus considering profiles belonging to the same user only); this is necessary for calculating precision and recall.
- The hash set of all profiles belonging to the users in the set; this cache guarantees fast lookup of a profile id to discover whether is contained in the set.
- A linear list containing a snapshot of the contained users. Users are stored primarily in a set to provide fast containment queries (is a user in this set?). The snapshot guarantees a consistent order of the users in the set in between modifications; that is, until a user is added or merged with the set, users can be listed in the same order.

Cache invalidation is performed every time a modification occurs, which means during addition of an user or merging of a user due to the acquisition of a matched pair. Cache rebuilding only takes place during the first read after an invalidation.

Parallelization

Parallelization in the learning algorithms or in post-processing has not been investigated. There is instead a much stricter bottleneck that has to be addressed: retrieval of the users details from the platforms of interest.

There are two different issues at play in the retrieval process.

First, network calls add considerable latency. For this reason every profile is retrieved first and kept for long-term storage into a local database.

Second, rate limits impact the availability of the platforms API, which may stop the application from issuing too many HTTP requests in a given period of time.

At the time of this writing, these are the limits imposed by the platforms:

- Facebook allows 600 calls every 600 seconds per seed (Quora [27])
- Twitter allows 350 requests per hour per seed (Developers [9])
- LinkedIn provides from 50 to 300 requests per seed as a daily limit, plus a 100k application limit (Network [25])

Each seed corresponds to a different authentication token: given the nature of the rate limits, it is possible to parallelize data retrieval by issuing many simultaneous requests but at the same time creating only one process per each token. Moreover, it is possible also to parallelize over the platforms axis as different platforms do not interact with each other and bandwidth is far from be saturated.

Thus a number of different Spy objects run in parallel, each instantiated with a seed and sending its results to a single database, where users are inserted as new documents.

Each Spy is able to detect rate limits notices issued by the APIs by inspecting HTTP response headers or fields of the JSON response bodies. In case a rate limit is detected, the Spy waits for 15 minutes before the next retry, but does not block the other processes.

4.6.2 Testing

Automated testing of the application (with developer-oriented test) is carried out by a mixture of integration and unit tests.

Unit tests cover the inner workings of the application, by exercising objects in isolation or in collaboration with other in-memory objects. Integration tests cover the interaction with external systems like the MongoDB database and the platforms APIs as suggested in Freeman and Pryce [13].

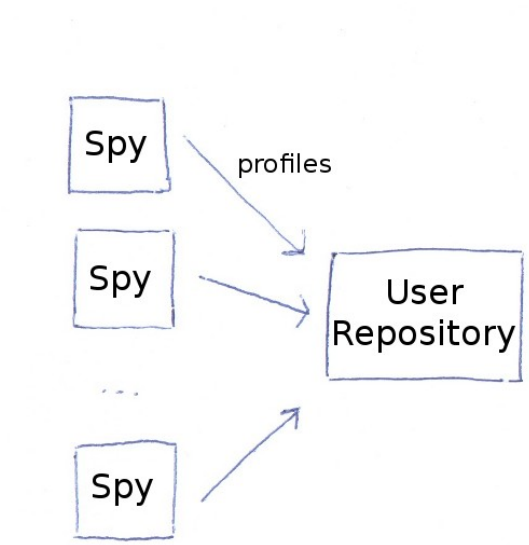

Figure 4.5: Spy objects running in parallel on each seed

It is particularly important to be able to execute automated tests against external APIs to guarantee an early detection of interaction problems in case the API is updated or changed in a way that interrupts the extraction of profiles or of some of the necessary fields.

Integration testing of the platform profiles extraction layer is then performed with a set of stored tokens and in some cases even with fake accounts.

For what regards authorization tokens, they are generally long-lived and with no expiration time; for this reason they can be included in automated tests without a particular need for maintenance. An exception is Facebook, which changed its policy to limit the duration of tokens to 2 months; the token used by the suite needs then to be renewed periodically.

Reliable test repeatability can also be ensured by targeting the fake accounts for extraction where necessary. For example, the profile of the Facebook user

$$JohnnyGalt[https://www.facebook.com/johnny.galt.733]$$

is extracted during the automated tests for the Facebook platform integration. Since no real person is regularly updating the profile, the test can perform strict assertion

on its content - real name, birth data, homepage and other fields. The test is thus able to fail when the platform changes its API in a non backward compatible way or when a regression is introduced in the Facebook-related code.

```
1  @Test
2  public void aFriendMembershipIsExtracted() throws JSONException {
3      UserAccess johnGaltAccess = access.getRelationship("Johnny
           Galt");
4      JSONObject membership = johnGaltAccess.getMembership();
5      assertTokenIsPresent(platform.getToken().toString(),
           membership);
6      assertFieldEquals("platform", "facebook", membership);
7
8      JSONObject profile = membership.getJSONObject("
           platform_profile");
9      assertFieldEquals("id", "100003792992839", profile);
10     assertFieldEquals("profile_url", "http://www.facebook.com/
           johnny.galt.733", profile);
11     assertFieldEquals("nickname", "johnny.galt.733", profile);
12     assertFieldEquals("date_of_birth", "1950-01-01", profile);
13 }
```

5

Conclusions and future work

This work has applied known record linkage and machine learning techniques in the domain of online social networks, demonstrating that accurate results can be achieved in the automatic alignment of profiles belonging to a unique person.

The experimental results in the comparison of models and post-processing strategies have shown that:

- There is a significant advantage in performing a post-processing step (i.e. the *single profile per platform* strategy), for every given technique used for classifying pairs, with improvements in precision of up to 16%.
- Decision trees can achieve an higher accuracy in this domain than linear SVMs, and in particular an higher recall.
- Nevertheless, linear models produced by support vector machines reach a good level of accuracy but maintain one order of magnitude faster training times.

There are several directions that further research in the social network domain may follow.

For example, the recollection of other real datasets or the generation of synthetic ones could help the comparison of classifiers on a larger sample. Given the moderate amount of seeds necessary to generate the dataset presented in this work, it is definitely possible to collect a larger sample, although the production of ground truth will be time expensive.

Datasets can not only be expanded by adding profiles from the existing platforms, but also by integrating new platforms into the process.

An orthogonal direction for further research is the introduction of other approaches for classification, such as boosting or bagging of decision trees, entirely different tree models or classifiers. Support vector machines have a choice of kernel methods that can be evaluated for studying non-linear models.

Performance optimization is possible in many phases of the reconciliation process: for example, tweaking windowing could provide a nearly linear speedup. A sensitivity analysis on the existing similarities may exclude the ones that are not useful for classification or that are already represented by correlated features.

A final direction could be the increase of accuracy with different features; there are two kind of features which haven't been used in this work and whose effect on precision and recall is yet to be discovered.

The first of these features is the information from the social graph: friendships and connections of the profiles to pages and groups. It can be expected that social connections such as people already identified on multiple platforms can be helpful in identifying new users, just as seed users are.

The second of these features is of a multimedia nature. For instance, avatars and photographs representing the profiles from multiple platforms could be compared, in a form of face recognition that attempts to conserve high accuracy when when information with a standard representation (e.g. names and birth dates) are hidden, missing or inaccurate.

Bibliography

[1] . Tailor: A record linkage tool box. In *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, pages 17–, Washington, DC, USA, 2002. IEEE Computer Society. URL http://dl.acm.org/citation.cfm?id=876875.879014.

[2] Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, and Luo Si. Expertise retrieval. *Found. Trends Inf. Retr.*, 6(2–3):127–256, February 2012. ISSN 1554-0669. doi: 10.1561/1500000024. URL http://dx.doi.org/10.1561/1500000024.

[3] Sergey Bartunov, Anton Korshunov, Seung-Taek Park, Wonho Ryu, and Hyungdong Lee. Joint link-attribute user identity resolution in online social networks. 2012.

[4] Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 39–48, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956759. URL http://doi.acm.org/10.1145/956750.956759.

[5] Francesca Carmagnola and Federica Cena. User identification for cross-system personalisation. *Inf. Sci.*, 179(1-2):16–32, January 2009. ISSN 0020-0255. doi: 10.1016/j.ins.2008.08.022. URL http://dx.doi.org/10.1016/j.ins.2008.08.022.

[6] Francesca Carmagnola, Francesco Osborne, and Ilaria Torre. User data distributed on the social web: how to identify users on different social systems and collecting data about them. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 9–15, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0407-8. doi: 10.1145/1869446.1869448. URL http://doi.acm.org/10.1145/1869446.1869448.

[7] Peter Christen. Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 151–159, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1401913. URL http://doi.acm.org/10.1145/1401890.1401913.

[8] William W. Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 475–480, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775116. URL http://doi.acm.org/10.1145/775047.775116.

[9] Twitter Developers. Rate Limiting | Twitter Developers . https://dev.twitter.com/docs/rate-limiting, 2012. [Online; accessed 07-09-2012].

[10] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923, 1998.

[11] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[12] Piero Fraternali, Marco Tagliasacchi, Davide Martinenghi, Alessandro Bozzon, Ilio Catallo, Eleonora Ciceri, Francesco Nucci, Vincenzo Croce, Ismail Sengor Altingovde, Wolf Siberski, Fausto Giunchiglia, Wolfgang Nejdl, Martha Larson, Ebroul Izquierdo, Petros Daras, Otto Chrons, Ralph Traphoener, Bjoern Decker, John Lomas, Patrick Aichroth, Jasminko Novak, Ghislain Sillaume, F. Sanchez Figueroa, and Carolina Salas-Parra. The

cubrik project: human-enhanced time-aware multimedia search. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 259–262, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1230-1. doi: 10.1145/2187980.2188023. URL http://doi.acm.org/10.1145/2187980.2188023.

[13] Steve Freeman and Nat Pryce. *Growing Object-Oriented Software, Guided by Tests*. Addison-Wesley Professional, 1st edition, 2009. ISBN 0321503627, 9780321503626.

[14] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm, 1996.

[15] Karl Goiser and Peter Christen. Towards automated record linkage. In *Proceedings of the fifth Australasian conference on Data mining and analystics - Volume 61*, AusDM '06, pages 23–31, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc. ISBN 1-920682-41-4. URL http://dl.acm.org/citation.cfm?id=1273808.1273812.

[16] Lifang Gu, Rohan Baxter, Deanne Vickers, and Chris Rainsford. Record linkage: Current practice and future directions. Technical report, CSIRO Mathematical and Information Sciences, 2003. URL http://festivalofdoubt.uq.edu.au/papers/record_linkage.pdf.

[17] Mauricio A. Hernández and Salvatore J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Min. Knowl. Discov.*, 2 (1):9–37, January 1998. ISSN 1384-5810. doi: 10.1023/A:1009761603038. URL http://dx.doi.org/10.1023/A:1009761603038.

[18] Tereza Iofciu, Peter Fankhauser, Fabian Abel, and Kerstin Bischoff. Identifying users across social tagging systems. In Lada A. Adamic, Ricardo A. Baeza-Yates, and Scott Counts, editors, *ICWSM*. The AAAI Press, 2011. URL http://dblp.uni-trier.de/db/conf/icwsm/icwsm2011.html#IofciuFAB11.

[19] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[20] Liang Jin, Chen Li, and Sharad Mehrotra. Efficient record linkage in large data sets. In *Proceedings of the Eighth International Conference on Database Systems for Advanced Applications*, DASFAA '03, pages 137–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1895. URL http://dl.acm.org/citation.cfm?id=789081.789250.

[21] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008. ISBN 978-0-521-86571-5.

[22] Alvaro E. Monge and Charles P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Proceedings of the SIGMOD 1997 Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 23–29, Tuscon, AZ, May 1997.

[23] Felix Naumann and Melanie Herschel. *An Introduction to Duplicate Detection*. Morgan and Claypool Publishers, 2010. ISBN 1608452204, 9781608452200.

[24] Mattis Neiling. Identification of real-world objects in multiple databases. In Myra Spiliopoulou, Rudolf Kruse, Christian Borgelt, Andreas Nürnberger, and Wolfgang Gaul, editors, *GfKl*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 63–74. Springer, 2005. ISBN 978-3-540-31313-7. URL http://dblp.uni-trier.de/db/conf/gfkl/gfkl2005.html#Neiling05.

[25] LinkedIn Developer Network. Throttle Limits | LinkedIn Developer Network. https://developer.linkedin.com/documents/throttle-limits, 2012. [Online; accessed 07-09-2012].

[26] R. Quinlan. Bagging, boosting, and C4. 5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 725–730, Menlo Park, August4–8 1996. AAAI Press / MIT Press. ISBN 0-262-51091-X.

[27] Quora. What's the Facebook Open Graph API rate limit? - Quora. http://www.quora.com/Whats-the-Facebook-Open-Graph-API-rate-limit, 2012. [Online; accessed 07-09-2012].

[28] Elie Raad, Richard Chbeir, and Albert Dipanda. User profile matching in social networks. In *Proceedings of the 2010 13th International Conference on Network-Based Information Systems*, NBIS '10, pages 297–304, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4167-9. doi: 10.1109/NBiS.2010.35. URL http://dx.doi.org/10.1109/NBiS.2010.35.

[29] Steven Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1:317–327, 1997.

[30] Titus Schleyer, Heiko Spallek, Brian S. Butler, Sushmita Subramanian, M. Louisa Poythress, Phijarana Rattanathikum, and Gregory Mueller. Requirements for expertise location systems in biomedical science and the semantic web.

[31] Cong Tang, Keith Ross, Nitesh Saxena, and Ruichuan Chen. What's in a name: a study of names, gender inference, and gender behavior in facebook. In *Proceedings of the 16th international conference on Database systems for advanced applications*, DASFAA'11, pages 344–356, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-20243-8. URL http://dl.acm.org/citation.cfm?id=1996686.1996731.

[32] Vassilios S. Verykios, Ahmed K. Elmagarmid, and Elias N. Houstis. Automating the approximate record-matching process. *Inf. Sci. Inf. Comput. Sci.*, 126(1-4):83–98, July 2000. ISSN 0020-0255. doi: 10.1016/S0020-0255(00)00013-X. URL http://dx.doi.org/10.1016/S0020-0255(00)00013-X.

[33] Jan Vosecky, Dan Hong, and Vincent Y. Shen. User identification across social networks using the web profile and friend network. *IJWA*, 2(1):23–34, 2010.

[34] Stefan Wehrli. Personality on social network sites: An application of the five factor model. ETH Zurich Sociology Working Papers 7, ETH Zurich, Chair of Sociology, 2008. URL http://EconPapers.repec.org/RePEc:ets:wpaper:7.

[35] Reza Zafarani and Huan Liu. Connecting corresponding identities across communities. In Eytan Adar, Matthew Hurst, Tim Finin, Natalie S. Glance, Nicolas Nicolov, and Belle L. Tseng, editors, *ICWSM*. The AAAI Press,

2009. ISBN 978-1-57735-421-5. URL http://dblp.uni-trier.de/db/conf/icwsm/icwsm2009.html#ZafaraniL09.