NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

POSTGRADUATE STUDIES

MASTER THESIS

Similarity-based User Identification across Social Networks

Aikaterini Zamani

Supervisors: **George Paliouras**, Researcher
Dimitrios Vogiatzis, Researcher
Stamatopoulos Panagiotis, Assistant Professor

ATHENS

NOVEMBER 2015

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ταυτοποίηση Χρηστών βασισμένη στην Ομοιότητα μέσω Ιστοχώρων Κοινωνικής Δικτύωσης

Αικατερίνη Χ. Ζαμάνη

Επιβλέποντες: **Γεώργιος Παλιούρας**, Ερευνητής
Δημήτριος Βογιατζής, Ερευνητής
Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΝΟΕΜΒΡΙΟΣ 2015

MASTER THESIS

Similarity-based User Identification across Social Networks

Aikaterini Zamani

R.N.: M1268

SUPERVISORS:

Paliouras George, Researcher
Vogiatzis Dimitrios, Researcher
Stamatopoulos Panagiotis, Assistant Professor

EXAMINATION COMMITEE:

Stamatopoulos Panagiotis, Assistant Professor NKUA
Koubarakis Manolis, Professor NKUA

ATHENS
NOVEMBER 2015

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ταυτοποίηση Χρηστών βασισμένη στην Ομοιότητα μέσω Ιστοχώρων Κοινωνικής Δικτύωσης

Αικατερίνη Χ. Ζαμάνη

Α.Μ.: Μ1268

ΕΠΙΒΛΕΠΟΝΤΕΣ :

Γεώργιος Παλιούρας , Ερευνητής
Δημήτριος Βογιατζής, Ερευνητής
Παναγιώτης Σταματόπουλος, Επίκουρος Καθηγητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Σταματόπουλος Παναγιώτης, Επίκουρος Καθηγητής
Κουμπαράκης Μανόλης, Καθηγητής

ΑΘΗΝΑ
ΝΟΕΜΒΡΙΟΣ 2015

Abstract

In this thesis we study the identifiability of users across social networks, with a trainable combination of different similarity metrics. This application is becoming particularly interesting as the number and variety of social networks increase and the presence of individuals in multiple networks is becoming commonplace. Motivated by the need to verify information that appears in social networks, as addressed by the research project REVEAL (REVEALing hidden concepts in Social Media), the presence of individuals in different networks provides an interesting opportunity: we can use information from one network to verify information that appears in another. In order to achieve this, we need to identify users across networks. We approach this problem by a combination of similarity measures that take into account the users' affiliation, location, professional interests and past experience, as stated in the different networks. We experimented with a variety of combination approaches, ranging from simple averaging to trained hybrid models. Our experiments show that, under certain conditions, identification is possible with sufficiently high accuracy to support the goal of verification.

SUBJECT AREA: Artificial Intelligence

KEYWORDS: user identification, similarity learning, entity resolution, machine learning, duplicate accounts

ΠΕΡΙΛΗΨΗ

Σε αυτή τη διπλωματική μελετάμε την ταυτοποίηση των χρηστών στα κοινωνικά δίκτυα, εκπαιδεύοντας εναν συνδυασμό διαφορετικών μετρικών ομοιότητας. Αυτή η εφαρμογή γίνεται ιδιαίτερα ενδιαφέρουσα, καθώς η αύξηση του αριθμού και της ποικιλομορφίας των κοινωνικών δικτύων και η παρουσία των ατόμων σε πολλαπλά δίκτυα γίνεται πλέον κοινός τόπος. Έχοντας ως κίνητρο την ανάγκη να επαλήθευσουμε τις πληροφορίες που εμφανίζονται σε κοινωνικά δίκτυα, όπως μελετάται στο ερευνητικό πρόγραμμα REVEAL (REVEALing hidden concepts in Social Media), η παρουσία ατόμων σε διαφορετικά δίκτυα παρέχει μια ενδιαφέρουσα ευκαιρία: μπορούμε να χρησιμοποιήσουμε τις πληροφορίες από ένα δίκτυο για να επαληθεύσουμε τις πληροφορίες που εμφανίζονται σε ένα άλλο. Για να επιτευχθεί αυτό, χρειάζεται να ταυτοποιήσουμε τους χρήστες σε διαφορετικά δίκτυα. Προσεγγίζουμε αυτό το πρόβλημα συνδυάζοντας κάποια μέτρα ομοιότητας που λαμβάνουν υπόψη τον εργασιακό χώρο, την τοποθεσία, τα επαγγελματικά ενδιαφέροντα και την εμπειρία των χρηστών, όπως αναφέρονται και καθορίζονται στα διάφορα δίκτυα. Έχουμε πειραματιστεί με μια ποικιλία από συνδυαστικές προσεγγίσεις, που κυμαίνονται από την απλή κατά μέσο όρο ταξινόμηση έως υβριδικούς εκπαιδευόμενους ταξινομητές. Τα πειράματά μας δείχνουν ότι, υπό ορισμένες προϋποθέσεις, η ταυτοποίηση χρηστών είναι δυνατή με αρκετά υψηλή ακρίβεια για να επιτευχθεί ο στόχος της επαλήθευσης των πληροφοριών.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: ταυτοποίηση χρηστών, μάθηση ομοιότητας, μηχανική μάθηση, διπλότυποι λογαριασμοί

ΕΥΧΑΡΙΣΤΙΕΣ

Πρωτίστως θα ήθελα να ευχαριστήσω θερμά τους ερευνητές κ.Γεώργιο Παλιούρα και κ.Δημήτριο Βογιατζή για την επίβλεψη της παρούσας εργασίας, καθώς επίσης και για την καθοδήγηση και τη βοήθεια που μου προσέφεραν όλο αυτό το διάστημα. Επίσης, αισθάνομαι την υποχρέωση να ευχαριστήσω τον καθηγητή του Τμήματος Πληροφορικής και Τηλεπικοινωνιών ΕΚΠΑ Παναγιώτη Σταματόπουλο, που μου έδωσε την ευκαιρία να συνεργαστώ με το ερευνητικό κέντρο ΕΚΕΦΕ ¨Δημόκριτος¨ για την εκπόνηση της διπλωματικής μου εργασίας. Τέλος, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στην οικογένεια και τους φίλους μου για την πολύτιμη συμπαράσταση και στήριξη που μου προσέφεραν όλα αυτά τα χρόνια.

LIST OF PUBLICATIONS

Zamani Katerina, Georgios Paliouras, and Dimitrios Vogiatzis. “Similarity-Based User Identification Across Social Networks." Similarity-Based Pattern Recognition. Springer International Publishing, 2015. 171-185.
The final publication is available at http://dx.doi.org/10.1007/978-3-319-24261-3_14 .

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Problem Description

Social network services have become part of our everyday life. It is now commonplace that people have accounts in multiple social networks, sharing their thoughts, promoting their work and probably influencing a part of the population via them. A variety of functionalities are provided by these services, such as video and photo uploading, posting, messaging, republishing etc, differing according to the platform and its aim.

Motivated by the need to verify the validity and trustworthiness of information that appears on social networks, the presence of individuals in different networks can be proved particularly useful. Public information from one network can be used to validate the source of information in another network. To achieve this goal, there is a need for user identification across social networks.

The trustworthiness of information in social networks, according to the REVEAL project [1], can be assessed on the basis of three pillars: Contributor, Content and Context, themselves supported by various modalities that are organized in two levels. The first-level modalities, such as reputation, presence, influence etc, are calculated directly from social media data, while the modalities of the second level, such as trustworthiness, misbehavior etc, rely on the results of the first level [1]. Our study contributes to the presence modality of REVEAL as user identification provides information about individuals in different platforms. Our research relies on Reveal's Journalism scenario, where verification and validation of information sources are essential.

In this study, we focus on individuals who are interested in promoting their professional activities in social media. We assume that these individuals often provide their real name in different social networks and therefore, the problem that we need to solve is primarily that of name disambiguation. Specifically, our approach compares users who have similar names, based on public information provided by the users, as returned by the search engine of the respective network. In other words, starting with the account of a user in one social network SN_1 we want to identify the account of a user with a similar name in another social network SN_2, that most likely belongs to the same user.

In our study we try to identify users across two popular social networks: Twitter and LinkedIn. We experiment on these networks because they are both used regularly, though

[1] `http://revealproject.eu/`

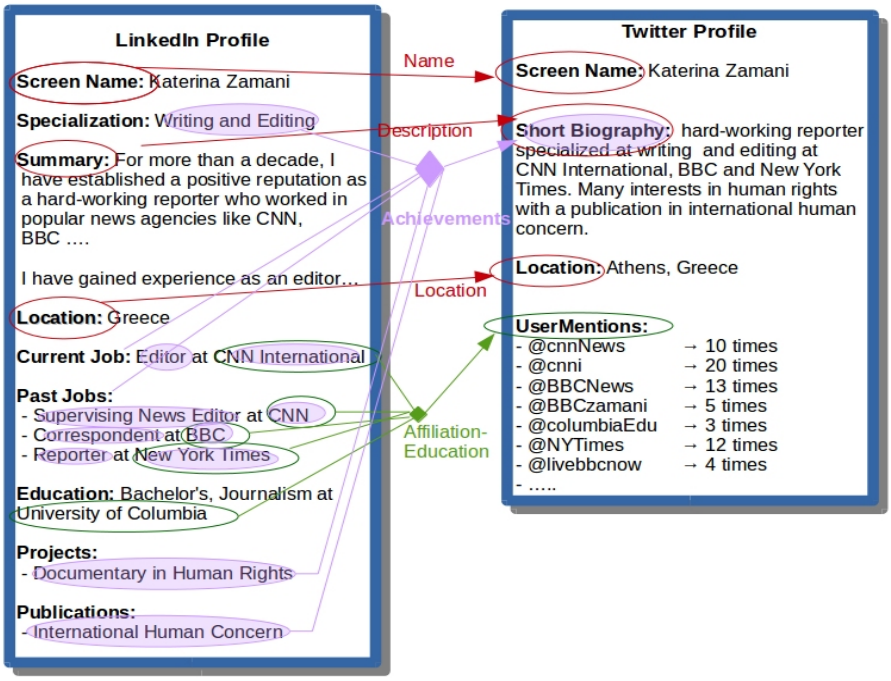

Figure 1.1: Example profiles and the alignment of their attributes, as used in the similarity metrics.

not exclusively, for professional purposes. Focusing on the journalism scenario of REVEAL, we form our target group of well-known news professionals.

Within a social network, each user is represented by a set of attributes that forms the user's profile. We derive a subset of these attributes based on the public accounts of users in the respective network. The LinkedIn profile of a user includes the following attributes: screen name, summary, location, specialization, current/past jobs with the respective affiliations, education, as well as projects and publications. On the other hand, the Twitter profile of a user contains: screen name, short biography, location and the other users, that the user specifies in her tweets (user mentions). Although the process starts with a name search, screen name can be considered as a feature because the results of the search engine do not always fit exactly the query. Figure 1.1 presents a simple example of how the user's attributes are aligned in the two networks, in order to be used in the similarity metrics. Some attributes can be aligned in a straightforward manner, e.g. Name, Location and Description, while others require the combination of various fields in the original user profile, e.g. Achievements and Affiliation-Education.

1.2 Contribution of our Work

In this study, we try to identify users across two popular networks: LinkedIn and Twitter. Our approach relies on novel similarity measures that mainly take into consideration professional information about the users. To achieve a satisfactory combination of the proposed similarity metrics, we experiment with various supervised classification techniques, such as decision trees, Naive Bayes, KNN and two hybrid classifiers that merges Naive Bayes and decision tables or decision trees efficiently. In addition, an attempt is made to deal with the imbalanced data problem and estimate the value of missing fields. Experiments based on a real world scenario show that highly accurate in user identification between these networks is indeed possible. Thus, the main contribution of our work is to show that the proposed approach of combining different similarity metrics is a viable solution to the identification of users, which in turn can be used to verify the validity of public information in social networks.

1.3 Structure of the Study

The remainder of the study is organized as follows.

In Section 2, we present closely related work and background algorithms, that are used in the similarity metrics.

In Section 3, we describe our approach to the problem. We present the similarity metrics that we used and we give specific examples of how they have been used to our approach. Finally, we describe the classification techniques that we have used to combine these metrics.

In Section 4, we describe the dataset that we used to train and validate our method. We also present the set up of our experiments and focus on the procedure for handling missing fields, an important problem in profile comparison. Then we mention the evaluation metrics that we have used to measure the performance of the classification techniques and we analyze our experimental results.

Finally in Section 5, we summarize the main conclusions and propose possible extensions of our work.

2. RELATED WORK

2.1 Related Studies

Various recent studies focus on the problem of user identification across the web. To the best of our knowledge this is the first study that is motivated by the verification of the validity of information based on public professional data provided by users in social networks. The novelty of this approach lies in the combination of different sets of features, that are associated to the professional aspects of a user account, in order to validate professional accounts.

The user identification problem is related to record linkage and duplicate detection in databases. Elmagarmid, Ipeirotis and Verykios [2] analyze extensively different similarity measures and efficient techniques for duplicate record detection. Specifically their survey contains two main parts. The first part presents a variety of simple and fast string matching techniques and the particular types of errors that each technique addresses. The second part involves more sophisticated machine learning techniques, such as probabilistic inference models, that match complicated records with multiple fields. As these techniques can detect nonidentical duplicate entries in databases, they can be particularly useful in the user identification problem. A similar study is presented by Cohen et al. [3]. They introduce the architecture of the "secondString" toolkit, which includes many well-known metrics for named entity matching. The matching of the entities is based on the computation of a similarity function, according to different metrics. The similarity function produces a real number, which indicates how similar or dissimilar are the two entities.

Additionally, many approaches have been proposed for correlating specifically social network accounts by exploiting information that is either explicitly or implicitly provided by the users. Explicit feedback refer to data that the users provide directly to their profile (e.g. during their registration). On the other hand implicit feedback refer to other data in the profile of a user (e.g.tags, posts etc) [4]. For example Vosecky, Hong, and Shen [5] combine different explicit profile fields by composing comparison vectors. Their work focuses on vector-based comparison algorithms for user profiles in social networks. In addition, Iofciu et al. [4] study the influence of tags in user identification across tagging network services relying on the combination of implicit and explicit information. They present an approach that takes into consideration the tags associated with a profile and aggregates profiles from different tag-based social networks. Therefore, they only deal with tag-based social networks.

Malhotra et al. [6] utilizes explicit feedback in order to model the digital footprints of users. Their study deals with profile disambiguation in social networks and focuses on LinkedIn and Twitter, like our work. Malhotra et al pair the accounts from the different social networks and they derive a similarity feature vector for each pair. This vector consists of similarity scores, by comparing the well-known comparable profile fields such as name, description, location, image and number of connections. They evaluate their approach with the use of widely used classifiers. Their work is the one that comes closest to our approach, but it also bears a number of differences from it. Firstly, due to our original motivation of verifying the validity of professional accounts, we focused on a different set of features to be extracted from the user profiles , i.e. we combine features that provide professional information about the users. Also, Malhotra et al. handle differently the problem of imbalanced data. Namely they use random sub-sampling to balance the training data, thus training their model with the same number of match and mis-match examples. Finally, our work addresses the issue of missing feature values, which is not dealt with in Malhotra et al. [6].

The authors of [7] focus on the use of implicit features of a user's activity, such as location, timestamps and writing style. Such features are particularly common in activity-based social networks. Specifically they take advantage of implicit information that users exhibit in their posts, such as geo-tagging, that specifies the user's location, the daytime of posting and textual characteristics that uniquely identify an individual. The aim of their study is to prove the possibility of tracking down a user across social networks, not necessarily to one specific match. Due to this purpose, their approach does not provide exact profile matches but instead a small set of possible ones that probably belong to a specific user. Related to authors, the aggregation of these implicit features in a single-aggregated profile could achieve an online footprint of a user, which needs defense from attacks in order to achieve the purpose of their study. The authors of [7] utilize activity-based attributes which are not available in the public profiles of the users.

The most recent work of Goga et al. [8], that is also close to our work, correlates users across different and popular social networks in large scale. Their study is based on public feature extraction and the proposed similarity metrics deal with explicit information. Due to the large scale of the data, they present a classification strategy which tries to find a threshold to separate matching and mismatching pairs. They also suggest ways to deal with the missing and imbalanced data. Their study uses several social network services and tries to identify accounts between pairs of them. When this is not feasible, the authors suggest the use of a correlation chain, which links accounts between social networks with the help of an intermediate social network [8]. The main difference of our approach is based on our motivation of verifying the validity of professional accounts. For this reason, the similarity

metrics that they use are more simplified and do not include professional aspects of the users, as they provide in their accounts. In addition, their main contribution is the matching of user accounts across major social networks that are more "sensitive to attacks" due to their nature, such as Facebook, Twitter and Google+, thus LinkedIn service is not included in their experiments.

2.2 Background Algorithms

2.2.1 Jaro-Winkler distance metric

The Jaro-Winkler metric is based on the number and order of common characters between two strings. The computation of the original Jaro metric [2] between two strings –s_1 and s_2 –follows three steps:

1. Compute the lengths of the strings, $||s_1||$ and $||s_2||$.
2. Find their *matching characters*. We consider as matching characters those that are the same in the two strings and the difference in their sequence order does not exceed a threshold. This threshold is set to be half of the length of the shorter of the two strings.
3. Count the number of *transpositions*. The previous step produces two subsequences of matching characters, one for each corresponding string. The number of characters, that are in the same position in the subsequences and do not match, represents the number of transpositions.

The Jaro distance is defined by the following equation:

$$Jaro(s_1, s_2) = \frac{1}{3} \cdot \left(\frac{m}{||s_1||} + \frac{m}{||s_2||} + \frac{m - \frac{t}{2}}{m} \right) . \tag{2.1}$$

where m is the number of matching characters and t the number of transpositions.

In 1999 Winkler recommended a modification of Jaro's metric, in order to give preference to strings that match exactly at the beginning (given a prefix length). The Jaro-Winkler distance is defined as:

$$JaroWinkler(s_1, s_2) = Jaro(s_1, s_2) + \Big(p \cdot l \cdot \big(1 - Jaro(s_1, s_2)\big) \Big) , \tag{2.2}$$

where l is the length of common characters at the prefix and p is a weight that indicates the significance of a common prefix length. The maximum value of l is usually set to 4 characters, while p is usually set to 0.1 and should not be longer than 0.25.

2.2.2 Levenshtein distance metric

The Levenshtein metric belongs to the family of edit distance metrics. Generally, the edit distance between two strings is the minimum number of single-character edit operations (insertion, deletion, substitution), needed to convert one string to the other. In Levenshtein distance, the cost of every edit-operation is set to 1, therefore it gives the same weight to every character transformation.

2.2.3 Smith-Waterman metric

Smith-Waterman is a dynamic-programming algorithm that finds the optimal local sequence alignment. Specifically, viewing each string as a sequence of characters, it compares every segment of possible lengths of the two sequences, in order to identify local regions with high similarity i.e., substring matching. The algorithm tries to divide the problem into smaller parts and find their solutions. To the final solution of the entire problem is computed by combining the smaller solutions.

The Smith-Waterman algorithm uses a two-dimensional matrix in order to score every pair of single-characters, produced by the two sequences of the strings. The score depends on a scoring system, which defines the costs of the edit operations (insertion/deletion of a character, substitution of one character by another) and gap penalties (open or extend gap). The score of each pair in a matrix cell is based on the scores of the left, top and top-left neighboring cells in the matrix. Specifically, the score of a cell in the (i, j) position is:

$$C_{i,j} = max\big(C_{i-1,j-1} + score_{i,j},\ C_{i,j-1} + W_k,\ C_{i-1,j} + W_k,\ 0\big)\ , \tag{2.3}$$

where $score_{i,j}$ is the score of the current cell computed by the costs of the respective edit-operations that are needed. Also W_k represents the penalty of a gap alignment of k length.

Once the matrix is filled with the scores, the algorithm starts from the cell with the highest score and backtracks the path of cells in descending score order [9]. The path produces the optimal local alignment sequence, i.e the best substring match of the compared strings.

The Smith-Waterman algorithm is based on the NeedlemanWunsch technique. The NeedlemanWunsch technique is used for global alignment and compares the strings of same length. On the other hand, Smith-Waterman compares sequences of different length, using local

sequence alignment. Nevertheless, the main difference between the two algorithms is that Smith-Waterman does not accept negative scores. Therefore, the Smith-Waterman technique replaces negative scores with 0, as it is denoted in eq. 2.3, in order to achieve local alignment.

2.2.4 SoftTF-IDF metric

SoftTF-IDF belongs to the family of token-based similarity metrics, where sets of tokens are compared to produce a similarity score. This family of metrics is a good solution for comparing strings that have common words but in a different position or order. Although, these metrics solve the problem of word swaps, they are not flexible in spelling errors. Therefore it is necessary to combine them with another similarity metric to be used for pairs of tokens and compute a final score based on these values [10].

The basis of SoftTF-IDF is TF-IDF. TF-IDF is a weighting scheme that denotes the significance of a term in a corpus. Assume two strings that are converted into two corresponding token sets. TF measures the frequency of a token in the respective token set, while IDF measures the importance of the token into the combination of the two token sets. The rational behind IDF is to weigh less the words that have little importance in the token sets, as they appear many times in both sets, while giving a heavier weight to the the rare ones.

SoftTF-IDF combines TF-IDF with a sub-metric and takes advantage of the above privileges. Assume two strings and S_1 and S_2 their corresponding token sets. Let *CLOSE(θ, S_1, S_2)* be the set of tokens $w \in S_1$, where each w match with at least one token of S_2, achieving a similarity score above θ. Also let *N(w, S_2)* be the maximum of the similarity scores of w with each matching word in S_2. It is worth to mention that $CLOSE$ function is symmetric. Thus SoftTF-IDF follows the bellow equation:

$$SoftTFIDF(S_1, S_2) = \sum_{w \in CLOSE(\theta, S_1, S_2)} W(w, S_1) \cdot W(w, S_2) \cdot N(w, S_2) \tag{2.4}$$

where

$$W(x, X) = \frac{TFIDF(x, X)}{\sqrt{\left(\sum_{x \in X} TFIDF(x, X)^2\right)}} \tag{2.5}$$

In our approach we use the Jaro-Winker distance as a sub-metric (i.e. for the calculation of $CLOSE$), setting θ to 0.9.

3. APPROACH

3.1 Description of Profiles

As explained in Section 1.1, the basic idea of our approach is to pair accounts that result from name search and identify those that belong to the same user. Therefore, the task that we are dealing with is translated to a classification of account pairs into two classes: "match" and "mis-match".

Specifically, in order to identify users we create a similarity vector for each pair of user profiles. The representation of our similarity vector is based on the definition proposed by [5]. Suppose that we have two user profiles from different social networks:

$$u_1 \in SN_1 \quad and \quad u_2 \in SN_2 \; . \tag{3.1}$$

The similarity vector of the two profiles is defined as:

$$V(u_1, u_2) \; = < score_1, score_2, \cdots, score_n > \; . \tag{3.2}$$

where $score_k$ corresponds to the score, returned by the k$^{\text{th}}$ similarity metric.

In order to facilitate the comparison, the similarity scores are normalized in the range [0.0, 1.0].

In the following sections, we first present the similarity measures that we use and then the methods we tested for classifying similarity vectors.

3.2 Similarity Measures

In this subsection we describe the similarity metrics that we use, in order to construct the similarity vectors for pairs of user profiles.

3.2.1 Name Measure

Previous work in record linkage [3] recommend Jaro-Winkler as an appropriate similarity for short strings while a study in user identification [4] propose Levenshtein distance for username similarity. Therefore, in our approach we test both distances (Jaro-Winkler and Levenshtein distance) in order to find the similarity between the screen names of users –first

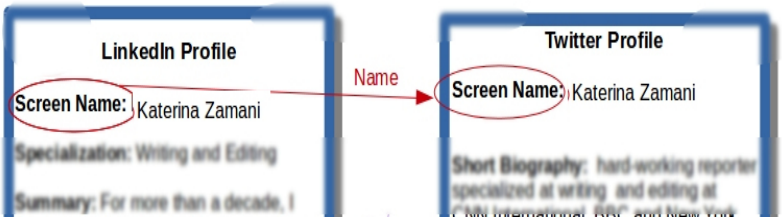

Figure 3.1: Example of Name measure.

and last name that a user provides during her registration. In Figure 3.1 you can see how the name attributes are aligned. Experimentally, the Jaro-Winkler metric has proved more accurate for our name measure. Due to the name ambiguity problem mentioned in Section 1.1, additional information is needed for user identification.

3.2.2 Description Measure

This measure estimates the similarity between the short biographies or summaries that users provide in different social networks, in order to describe themselves, their work and their specialization. An example is shown in Figure 3.2. In order to measure similarity according to this short description, we pre-processed corresponding fields of the two profiles. We removed the punctuation, lowercased and tokenized the description, thereby creating two different token lists. Taking into consideration the example in Figure 3.2, the two token lists are as follows:

A_1 =[for, more, than, a, decade, i, have, established, a, positive, reputation, as, a, hard, working, reporter, who, worked, in, popular, news, agencies, like, cnn, bbc, i, have, gained, experience, as, an, editor]

A_2 = [hard, working, reporter, specialized, at, writing, and, editing, at, cnn, international, bbc, and, new, york, times, many, interests, in, human, rights, with, a, publication, in, international, human, concern]

We tested two different approaches:

1. **JaccardToken similarity**: Comparing each pair of tokens in the two token lists, we determine the number of common words. The similarity of the description fields is the ratio of this number to the number of not common ones.

2. **WhitespaceGrams similarity**: We create a set that contains the union of tokens of the two token lists, without duplicates. By comparing the new list against each token list we determine the number of common tokens among the list and the token set. The

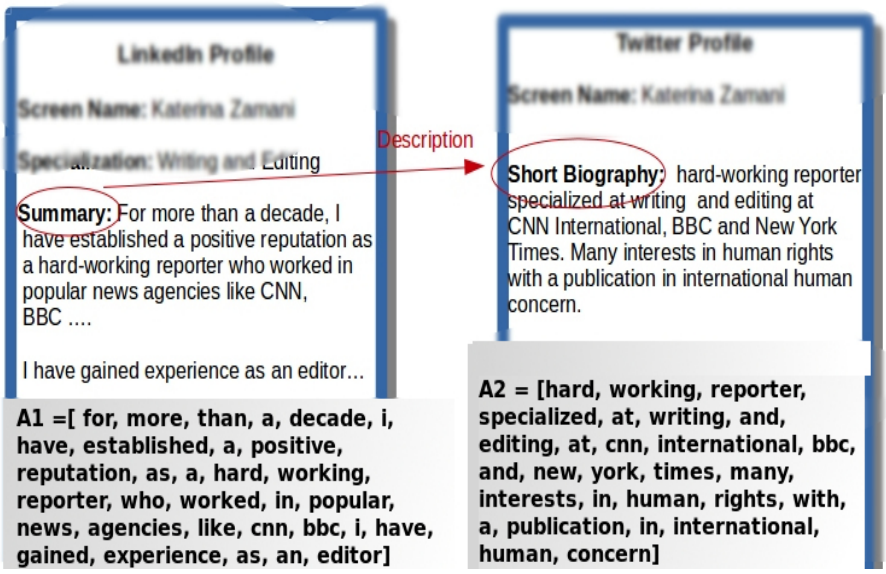

Figure 3.2: Example of Description measure.

difference of the two resulting numbers (one for each list) denotes the dissimilarity of the initial token lists if we take into account their non-common words. Therefore, the similarity of the description fields is computed by the following equation:

$$WG(s_1, s_2) = \frac{\|s_1\| + \|s_2\| - |n_1 - n_2|}{\|s_1\| + \|s_2\|} \,. \tag{3.3}$$

where n_1, n_2 represents the number of common tokens among the set of all tokens and each of the two token lists s_1 and s_2 respectively.

LinkedIn description field tend to be larger than Twitter's short biographies. Due to the disproportionate length of the description fields, there is a need to express the ratio of common words more accurately. Therefore, the *WhitespaceGrams similarity* approach seems more adequate in our problem. This is also confirmed experimentally.

3.2.3 Location Measure

A recent study associates location with the user's posts, based on attached geo-tags [7]. Although it is a promising approach, it is not directly applicable to all social networks, e.g. LinkedIn doesn't provide geo-tagging. Also it is common that users provide different variants of their locations in the social networks. For instance they may provide their country in one social network, but their city written in their native language in the second one. Techniques based on string similarity will fail in such situations. For this reason our

comparison combines the textual representation of the location field with geonames ontology [11] to obtain geospatial semantics. In particular, by querying the web service, we receive the geographical information of a place. If a given location does not exist in the ontology, the service identifies it as missing.

When we query the geonames ontology, we receive the results in a geospatial taxonomy first the countries, then the states, after the cities etc. Even when a name location corresponds to many toponyms, the ontology will sort the results in a geospatial way. Thus, we limit the results of a query to those containing the desirable location. When we query for two locations, that are provided in different social networks, we receive their geographical information and we compare only those locations that belong to the same country. We convert these locations to bounding boxes or to coordinates, depending on their geospatial relation. Specifically, the similarity score of the two locations is defined as follows:

1. The ratio of bounding box areas if one bounding box is within the other.
2. The Euclidean distance between their coordinates when the locations belong to the same country.
3. 0.0 otherwise.

Specifically, the above enumeration can be defined by the following equation:

$$LocSim(l_1, l_2) = \begin{cases} Bbox(l_1)/\ Bbox(l_2) & \text{if } Bbox(l_1) \subseteq Bbox(l_2) \\ Bbox(l_2)/\ Bbox(l_1) & \text{if } Bbox(l_2) \subseteq Bbox(l_1) \\ 1/\ (1 + \|l_1 - l_2\|_2) & \text{if } l_1,\ l_2\ in\ \text{SC} \\ 0.0 & \text{otherwise} \end{cases} \tag{3.4}$$

where *Bbox* represents the bounding box of the respective location and *SC* refers to the same country. We always put the bounding box that is subsumed into the denominator, in order to produce values in the range of [0.0, 1.0]. Nevertheless, the ratio of the bounding box areas would probably produce small values, due to huge disproportion of the respective covering areas. For this reason, we normalize the similarity score in logarithmic scale as follows:

$$Normalize(value) = \frac{1}{1 - \log_{10}(value)}\ . \tag{3.5}$$

Using eq. 3.5 , the similarity score in all situations takes values in the range [0.0, 1.0].

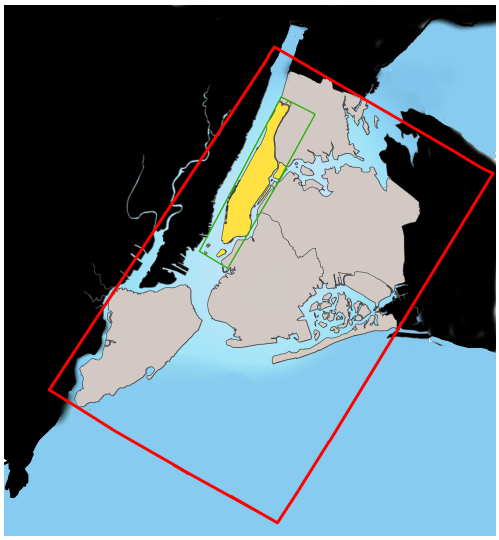

Figure 3.3: Example with bounding boxes in location measure.

Now lets show characteristic examples of the use of the location measure. First we assume that $SN_{location1}$= "New York" appears in one social network and $SN_{location2}$="Manhattan" appears in the other. Since Manhattan is a borough of New York City, its bounding box –imagine it as a rectangle that covers the Manhattan area –will be included into the bounding box of New York city. Thus, the similarity of the two locations is measured as the ratio between the covering area of Manhattan's bounding and the area of New York City's bounding box, as shown in Figure 3.3.

In the example of Figure 1.1, the similarity of the two locations will be 1.0 since the bounding boxes, which are returned from the ontology, coincide.

Now suppose that we retrieve two locations that belong to the same country but their bounding boxes are not subsumed –$SN_{location3}$="Athens" and $SN_{location4}$="Sparta". In this case, their similarity is computed by the Euclidean distance of the coordinates of the centres of two bounding boxes, as shown in Figure 3.4.

Figure 3.4: Compute location similarity using the Euclidean distance.

3.2.4 Affiliation-Education Measure

This measure attempts to match the current/past affiliation and educational experience of the users, as stated in the social network profiles. In order to measure the similarity score we create two token sets, one for each corresponding network. Figure 3.5 shows the profile fields that participate in this score. In LinkedIn's set we use the affiliation of current and past experiences and educational schools, while in Twitter's set we use the userMentions (@ symbol) that appear in the user's tweets. We assume that the user is likely to mention her affiliation and school names in her micro-blogging posts to promote her work. Neither of the two token sets include duplicates. The token set obtained from Twitter contains additionally the frequency of each userMention.

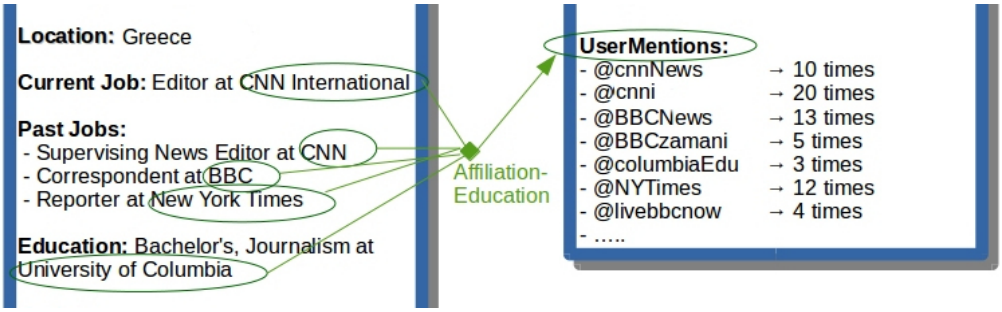

Figure 3.5: Example of Affiliation-Education measure.

An additional practical problem with userMentions is their appearance. Generally a userMention describes an entity in a small sequence of characters, in an abbreviated form and usually without the use of delimiters. It is also commonplace that some userMentions refer to the same entity with a different arrangement of the words, as you can see in Figure 3.5. Thus, there is a need for a textual comparison measure that is suitable for substring matching. Based on the related survey [2], the Smith-Waterman distance measure seems adequate, because it combines edit and affine gap distances to identify local regions with high similarity among them.

In particular we used the implementation of the measure from the simmetrics library [12]. We measure the similarity between each pair of tokens in the two token sets and keep only those similarity scores that exceed a predefined threshold t. Then we weigh the resulting scores according to the frequency of a userMention in Twitter profile. Therefore, the overall similarity score is calculated as shown in the following equation:

$$\sum_{i=1}^{n}(score_i \cdot freq_i) \,/\, \sum_{i=1}^{n} freq_i \tag{3.6}$$

where $score_i$ is the Smith-Waterman similarity score of a pair of tokens that is above the

threshold t and $freq_i$ is the frequency of appearance of the specific userMention in the user's tweets. The weight indicates a significance estimate of the corresponding userMention.

Some similarity scores that exceed the threshold may correspond to the same token of one of the sets. This is acceptable because many userMentions or jobs often refer to different variants of similar entities, as we explained above. For instance in Figure 3.6,"@BBCNews", "@BBCzamani" and "@livebbcnow" refer to related entities. Thus, the affiliation "BBC" in LinkedIn will be matched with all these entities and will lead to the aggregation of their scores.

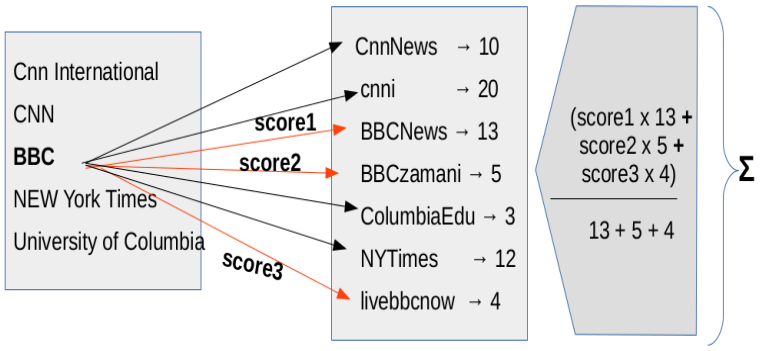

Figure 3.6: Example of computing Affiliation-Education measure.

3.2.5 Achievements Measure

It is common that users include their professional achievements and their job specialization in the short biography field of their profile. In fact we have observed that often the words a user provides in the description field in Twitter, can be matched with those that she provides for her job, publications etc in LinkedIn. We attempt to capture this by using the SoftTFIDF metric, which takes into consideration "similar" and not only identical tokens [2]. We compose a textual summary of the most significant professional achievements of a user, as she provides them in LinkedIn: we combine current and past job experiences and the corresponding affiliations, professional specialization, projects and publications that she has participated in. Figure 3.7 shows the profile fields that participate in the computation of the similarity score. The similarity between this "professional summary" and the short biography in Twitter is computed with the use of SoftTFIDF as implemented in the secondString library [13], [10].

SoftTFIDF converts the two texts into two bags-of-tokens and calculates the overall similarity score, by computing the TFIDF distance between each pair of tokens. The SoftTFIDF method prunes the token pairs if their Jaro-Winkler similarity is below 0.9. Due to the stop

words in the description field, the similarity score takes small values. For this reason we normalize the similarity score in logarithmic scale, with the use of eq. 3.5 .

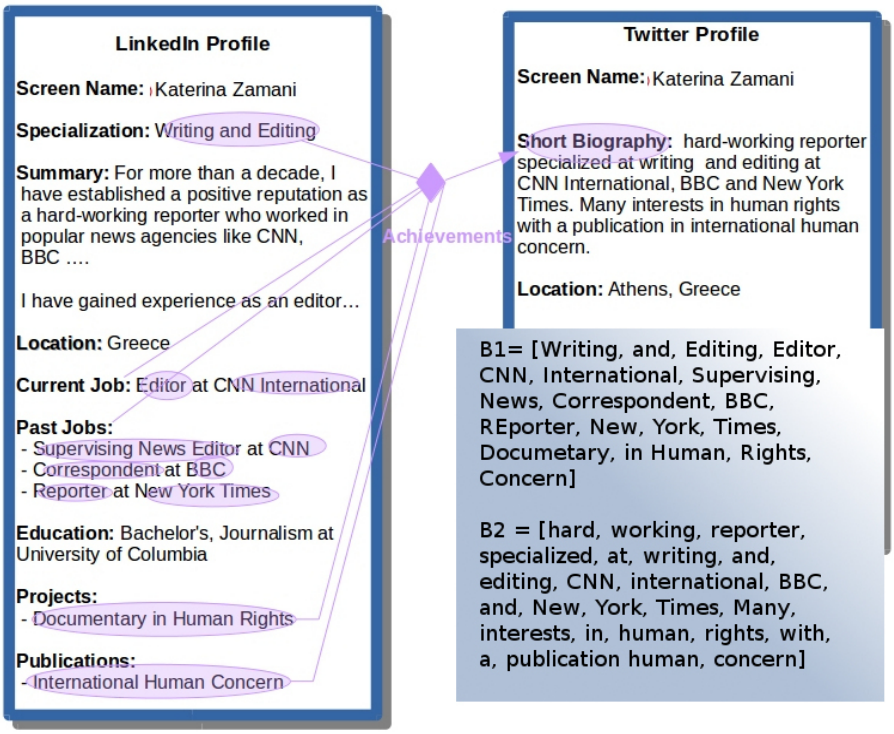

Figure 3.7: Example of Achievements measure.

3.3 Classification

As mentioned in Section 3.1, the various similarity measures are used to built similarity vectors. These vectors are then classified in order to achieve the required user identification. Below we describe the different classification approaches that we tested.

3.3.1 Baseline Classification Results

As a baseline we calculate the average of the scores in the similarity vectors:

$$AvgScore(V) = \frac{1}{n} \cdot \sum_{i=1}^{n} (score_i) \tag{3.7}$$

where $score_i$ corresponds to the respective score in the similarity vector.

As an example, assume the following user profiles:

$$u_1 \in SN_1 \quad and \quad u_2, u_3 \in SN_2 \ . \tag{3.8}$$

We pair the resulting profiles and two different similarity vectors are created.$V_1(u_1, u_2)$, $V_2(u_1, u_3)$ are as follows:

$V_1(u_1, u_2) = <1.0 , 0.345, 0.456, 0.678, 0.879 >$

$V_2(u_1, u_3) = <1.0 , 0.432, 1.0, 0.789, 0.654 >$

The baseline computes the following average scores :

$\text{AvgScore}(V_1) = 3.358 / 5 = 0.6716$

$\text{AvgScore}(V_2) = 3.875 / 5 = 0.775$

The higher the score, the more likely it is that the corresponding profiles belong to the same user. Since we are interested in a single match, the pair with the highest score is considered a match and all other are mismatches. In the Figure 3.8 we can see the the graphical representation of the above example. With the use of the baseline classifier, V_2 achieves a higher score. Therefore, $TUser2$ of the "Georgios Paliouras" Twitter set probably belongs to the "target user" $LUser1$ of the corresponding set of the LinkedIn dataset.

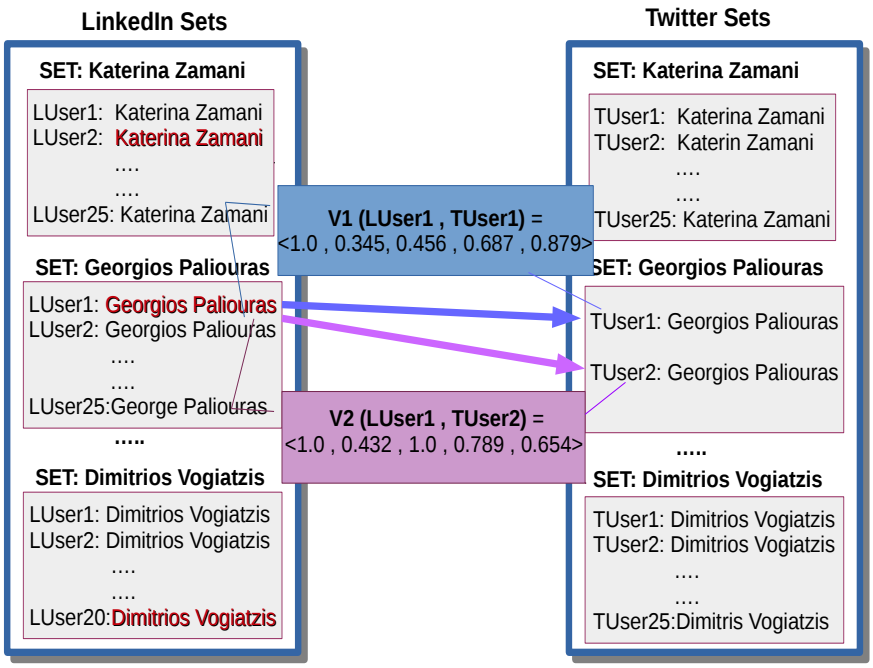

Figure 3.8: Example of Baseline Classifier in *LinkedIn identification* case. The red shadowed texts represent the target users.

3.3.2 Binary Classifiers

A different way to classify similarity vectors is by training binary classifiers to distinguish between matches and mis-matches.

The classifiers that we tested are presented below in brief:

Euclidean Distance:

This approach is based on the Euclidean distance of an instance from the centroid of each class. It classifies an instance based on the minimum of the distances of each center. This classifier serves as a baseline technique in our experiments.

The classification works as follows:

Assume a set of n classes and let $C = (c_1, c_2, \ldots, c_n)$ be the set of their centroids. Both the centroids, as well as the training and testing instances, are d-dimensional data points, where d is the number of attributes. The centroids are calculated on the training data. Assume $T_k = (t_1, t_2, \ldots, t_m)$ a set of training instances with class k. The value of centroid c_k is computed as the average score of each $t_i \in T_k$ in the d-dimensional space.

$$Center(T_k) = \frac{1}{M} \cdot \sum_{i=1}^{M} t_{i,j} \;\mid\; \forall\, j \in [1, \ldots, d] \;. \tag{3.9}$$

Now assume a testing instance $\bar{\mathrm{x}}$ as a d-dimensional data point. During testing, the classifier computes the Euclidean distance of $\bar{\mathrm{x}}$ with each centroid c_k. The class with the minimum Euclidean distance is chosen.

$$WinnerCenter(x, C) = min(\, EuclDist(x, c_k) \;\mid\; \forall\, c_k \in C) \;. \tag{3.10}$$

where

$$EuclDist(x, c_k) = \frac{1}{\|x - c_k\|_2} \;. \tag{3.11}$$

Thus, the label of $\bar{\mathrm{x}}$ is the class that is "closest" to $\bar{\mathrm{x}}$.

As we formulate the problem in binary classification, we assume a set of two classes, thus two centroids. Also the procedure takes place in a 5-dimensional space, due to the 5 scores in the similarity vectors. During the test phase, the classifier decides whether a pair is a match or not by determining the centroid that has the minimum Euclidean distance from the pair.

Decision Tree:

A common approach in supervised classification learning is that of decision-tree learning, which generates a decision tree as a predictive model. This model predicts the class of an instance, by traversing the decision-tree from the root to the leaves. Each non-leaf node (internal and root nodes) contains an attribute condition to separate an instance depending on its characteristics. Also it includes at least two branches, each one leading to a different node (internal or leaf) based on the decision that was produced from the condition. Having the decision tree model constructed, the classification of a test instance is straightforward. Starting from the root it follows the right path, until a leaf, according to the attribute values of the instance, which dictate the decision outcome of each node. A leaf node contains the predicted class label, based on the corresponding instance.

In our study we experiment with C4.5, a well-known algorithm in decision tree learning that can be used for classification. This method creates a decision tree model, by analyzing a training set of instances, in our case similarity vectors with the correct classification-labeling. Each internal node of the tree represents a test on one of the five similarity scores, that most effectively splits the respective set into more homogeneous subsets. The measure of the purity of each node is based on information entropy. Entropy is a measure that represents the amount of uncertainty in the current dataset –lower entropy denotes higher homogeneity/purity. Each leaf node of the tree holds a class label, corresponding to the majority class in the training instances. During the test phase, the classifier decides whether a pair is a match or not by traversing the decision model tree from root to the leaf. To avoid overfitting, we perform a common technique in decision trees, called pruning. Pruning removes some branches of the decision tree in order to improve its generalization capability [14].

Naive Bayes:

This classifier uses Bayes theorem, to combine prior knowledge of the classes with new test instances [15]. Specifically the classifier calculates the probability of a class given a test instance, assuming that all features of the vector, i.e. the five scores, are independent. As we use continuous features we calculate probabilities by fitting a Gaussian distribution [15]. In the training phase, the classifier estimates two probability distributions, one for each class. During testing, the classifier decides the label of a specific pair, depending on its probability of belonging to each of the two classes. The computation of these probabilities is based on the respective distributions that were estimated in the training phase.

KNN:

The Nearest-Neighbor classifier belongs to the category of lazy learning, because the training phase is performed only when the classification of the test instances is needed. This classifier represents each instance as a point in d-dimensional space, where d is the number of attributes, i.e. 5 in our case [15]. Lets assume a set of classes $C = (c_1, c_2, \ldots, c_k)$, a set N of d-dimensional training instances and x a d-dimensional test instance. KNN classifies the test instance (x) as follows:

1. It compares x with every training instance of the set N, computing its distance from each training instance, with the use of a proximity measure.
2. It finds the k nearest training instances of x and their classes respectively.
3. It chooses the label of x, based on the majority class among its k nearest neighbors.

In our study, we set the value of k to 5. Due to imbalanced of data in our dataset, we prefer to set an odd number in order to achieve a clear result in labeling. Experiments show that 5 performs better in our case. Moreover, we use the Euclidean distance as the proximity measure.

NBTree:

This is a hybrid approach that combines Naive Bayes and Decision Tree classifiers. This technique generates a decision tree as usual. The internal nodes of the tree contain splits triggered by single attributes, while the leaves represent Naive Bayes classifiers, thus returning continuous random variables instead of discrete classes. Before explaining the algorithm, we describe some fundamental notions:

- *Utility of a node*: The utility of a node is estimated by the 5-fold cross-validation accuracy of the NB classifier for that node.
- *Utility of a split*: The utility of a split is defined as a weighted sum of the utility of the nodes that are generated. The weight of a node is related to the number of instances that reach the node.

The algorithm starts by evaluating the utility of a split $U(x)$ on each attribute x. If an attribute is continuous, a threshold is found with the use of standard entropy minimization

technique [16]. Afterwards it chooses the attribute with the highest utility split and chooses that as a root node. The process is repeated recursively for each of the training subsets that are generated. In each level, the classifier decides whether to split the set, based on the significance of node utilities. If the accuracy of each Naive-Bayes classifier at each child node is higher than a single Naive-Bayes classifier at the current node, the set will be split [16]. Otherwise, it declares the current node as a leaf and creates a Naive-Bayes classifier for it.

DTNB:

This is another hybrid approach that combines Naive Bayes and Decision Tables classifiers. Decision Tables can be thought of as classification rules. Each row of the table represents a conjunctive rule and is associated with a class label [17]. Initially in DTNB, all features are modeled in Decision Tables. Afterwards with the use of forward selection, the classifier selects in a stepwise manner the features that improve a Naive Bayes classifier on a validation set the most. DTNB trains Naive Bayes on the selected features, while it constructs a Decision Table with the res of the features. The label of a test instance results from the combination of class probability estimations of both classifiers (Decision Table and Naive Bayes).

4. EXPERIMENTAL RESULTS

4.1 Data Collection

The collection of the data was based on name search, as denoted in Section 2. We started with a list of "target users" in mind, e.g. "Katerina Zamani". Each target user had a different name. Given the name of a particular target user, we gathered the first 25 profile results from each network, using the networks search engine. Thus, we created two sets of profiles (one for each network), each set containing the results of the search for a particular name. The aim of our study was to identify within each such set only the profile of the target user, given the users profile in the other network, e.g. given Zamanis profile in Twitter, we wanted to identify the profile of the same person in LinkedIn, among the set of profiles that the search for "Katerina Zamani" has returned. We did this matching both ways, i.e. from Twitter to LinkedIn and vice versa, but only for a single target-user with that name. This set-up is motivated by our goal of verifying the validity of profiles of professional individuals in social networks. We also assumed that each target-user has a single account in each network. Therefore, in each set we identified one profile as the correct match, while all others were considered mismatches. Figure 4.1 presents how we setup the data collection in corresponding sets, as explained in detain in this subsection.

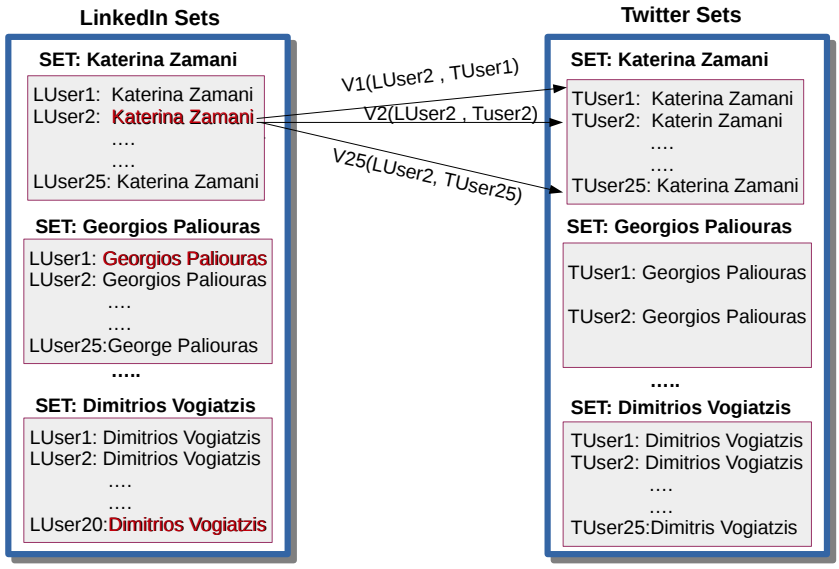

Figure 4.1: Data structure into corresponding sets. The red shadowed texts indicate the target users for each set, while the whole figure represents LinkedIn identification case.

In Table 4.1 the total number of profiles in each of the two networks that we used is provided. We separated the data into two datasets — one for each network. Each data set contained 262 profiles sets and each set included at most 25 profiles.

Table 4.1: Profiles in the datasets.

	LinkedIn	Twitter
Number of profiles	2766	3373
Number of profile sets	262	262

4.2 Experimental Setup

Starting with a profile from a social network (SN_1) and a set of profiles from a different one (SN_2), our aim is to find the profile that most likely matches the one from SN_1. In order to select the most likely match, we compare each profile in each set of SN_1, e.g. the set of "Katerina Zamani", with each profile of the corresponding set of SN_2. Each comparison produces a similarity vector, as described in Section 3.2, which is classified as a match or not. In our experiments we use two different datasets corresponding to the "direction" of the identification, i.e. starting with a profile from LinkedIn we compare it against the profiles of the corresponding set in the Twitter dataset and vice versa. Henceforth, we refer to the former task asTwitter identification and the latter as LinkedIn identification.

4.2.1 Missing Values

It is common that users do not complete each field of their profile. This influences the performance of our approach because many profile fields that we use are not available. Table 4.2 presents the number of missing fields for each similarity metric.

Table 4.2: Number of missing values.

SN/metric	Name metric	Description metric	Location metric	Affiliation-Education metric	Achievements metric
LinkedIn	0	1866	0	462	221
Twitter	0	1431	1582	735	1431

As shown in Table 4.2, the name of the user is never missing as it is a compulsory field during the user's registration. However, the location field in Twitter is only available for 53% of the users in our dataset. The availability of description/summary fields is 33% for LinkedIn and 58% for Twitter, while the userMention attribute in tweets is used by 78% of the users. Regarding Affiliation-Education and Achievements metrics, LinkedIn provided

more complete information than Twitter. This is due to the use of many fundamental and professional fields, such as affiliation, professional experience etc., that many users provide in their LinkedIn account.

4.3 Results for Separate Measures and Baseline Classifier

In this section we evaluate separately each similarity measure that we used. Taking into consideration the large percentage of missing values and how this could influence the accuracy of classification, we examined the following solutions:

- **Set a default score:** We set 0.5 as a default similarity score, when the score cannot be calculated. It is worth recalling that all scores are normalized in the range $[0.0, 1.0]$.
- **Set the average score:** We set the missing similarity score to the average value of the similarity scores, that can be computed from the available fields. This average score is different for each metric and it depends on the measured similarity scores of the respective measure.
- **Set the median score:** The basic idea of this approach is similar to the previous one, but instead of the average, we use the median value of the computed similarity scores.

In particular, we compute the recall of each similarity score separately. Note that precision is the same as recall here, since all methods are required to return exactly 262 matches. Specifically, we select as the most likely matching set the one with the maximum similarity score. Tables 4.3, 4.4 provide the results for the two datasets (*LinkedIn identification* and *Twitter identification*), for different missing values strategies.

As expected, the success scores in name metric are the same in all approaches because name fields are always available in social networks. If more than one pairs have the same maximum similarity score, we select as matching profile the one that is topper in the set of the returning results. However, the score in the *Twitter's identification* case is much higher, due to the different nature of the two search engines. On one hand the sequence of results that is returned from a search query in Twitter depends on the popularity of each account, while on the other hand LinkedIn's search engine categorizes differently its resulting user accounts. In addition, the high success scores of the two last metrics in the *LinkedIn's identification* case, indicate the importance of the professional fields in the identification. Finally we conclude that the average score approach to the in missing values problem, lead to better results. For this reason, we adopt this approach for the rest of our experiments.

Table 4.3: Recall for Linked identification for different measures and baseline classifier and for different strategies for missing values. Results are presented as percentages to facilitate readability.

Strategy for missing values	Default	Average	Median
Name measure	68.70%	68.70%	68.70%
Description measure	60.31%	64.12%	63.74%
Location measure	67.94%	69.08%	68.70%
Affiliation-Education measure	80.15%	79.77%	79.77%
Achievements measure	83.59%	**87.02%**	87.02%
Baseline Classifier	86.26%	86.64%	83.59%

Table 4.4: Recall for Twitter identification for different measures and baseline classifier and for different strategies for missing values. Results are presented as percentages to facilitate readability.

Strategy for missing values	Default	Average	Median
Name measure	90.84%	**90.84%**	90.84%
Description measure	80.92%	85.50%	85.50%
Location measure	75.57%	82.44%	79.78%
Affiliation-Education measure	75.19%	75.19%	74.43%
Achievements measure	74.81%	79.77%	79.77%
Baseline Classifier	74.81%	88.55%	85.11%

Also in this subsection, we assess the results of the simple average combination of the similarity measures as described in Section 3.2. For each profile set, we define as "match" the pair with the maximum average score. We use recall in this case as well, in order to measure performance.

For our experiments we utilize the databases of both social networks and we receive the recognition success, which shows how many ground-truth data identified correctly. We can conclude that this success corresponds to the recall measure of the positive class.

In the *LinkedIn identification* task and in the case of average missing values strategy, the simple combination recognizes correctly 227 pairs out of 262, arriving at a recall of 86.64%. In the *Twitter identification* task the respective recall is 88.55%. Although the results are promising for a simple combination, are somewhat lower than the best individual scores, i.e. the Achievement measure for LinkedIn (see Table 4.3) and the Name measure for Twitter (see Table 4.4).

4.3.1 Imbalanced Data

The nature of the identification problem across social networks results in considerable imbalance between the two classes (match vs. mis-match). In our study, only 9.5% of the LinkedIn profiles and 7.8% of the Twitter profiles comprise the minority (match) class. This imbalance can cause problems during training for some classifiers. In order to handle this issue, during testing phase we define as "match" the pair with the maximum probability, in reference to the classifier. The description and the results of the procedure are presented further below in Section 4.6.

4.4 Performance Measures

In order to present the results of the classification strategy, we will first describe the performance metrics that we use to evaluate the classification systems. As mentioned above, we use binary classifiers (match and mis-match). The evaluation metrics that are commonly used for this type of classifiers, are precision, recall and f-measure.

In order to present these measures, we enumerate the four potential results in a binary classification task:

- **True Positive (TP)**: the number of instances correctly predicted as belonging to the positive class.
- **True Negative (TN)**: the number of instances correctly predicted as belonging to the negative class.
- **False Positive (FP)**: the number of instances incorrectly predicted as belonging to the positive class.
- **False Negative (FN)**: the number of instances incorrectly predicted as belonging to the negative class.

Table 4.5, places these cases in a confusion matrix.

Table 4.5: Confusion Matrix.

Situations	Label "match"	Label "mis-match"
Classifier predicts "match"	True Positive (TP)	False Positive (FP)
Classifier predicts "mis-match"	False Negative (FN)	True Negative (TN)

4.4.1 Evaluation Metrics

The evaluation metrics that we used are based on the above four cases.

Precision:

Precision of a class C is the ratio of the correctly predicted instances to all the instances that the classifier retrieved, irrespective of whether they are correct or not [18]. Precision indicates what proportion of the instances that the classifier returned are correct. The precision of the positive and negative classes is shown in the following equations:

$$Precision(Positive) = \frac{TP}{TP + FP} \ . \tag{4.1}$$

$$Precision(Negative) = \frac{TN}{TN + FN} \ . \tag{4.2}$$

Recall:

Recall of a class C is the ratio of the correctly predicted instances to the total number of instances truly belonging to C. Recall measures the probability that the classifier would identify correctly the instances of the specific class C. The recall for the positive and negative classes is shown in the following equations:

$$Recall(Positive) = \frac{TP}{TP + FN} \ . \tag{4.3}$$

$$Recall(Negative) = \frac{TN}{TN + FP} \ . \tag{4.4}$$

Recall of the positive class is also called sensitivity or true positive rate.

Accuracy:

Accuracy is a statistical measure that indicates how well the classifier predicts overall. Accuracy is formalized in eq. /refeq:accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \ . \tag{4.5}$$

F-Measure:

Due to the imbalanced data, F-measure is more appropriate than accuracy to evaluate the overall classification performance of a method. F-measure represents the harmonic mean of precision and recall [18], as denoted in the following equations:

$$Fmeasure(Positive) = 2 \cdot \frac{Precision(Positive) \cdot Recall(Positive)}{Precision(Positive) + Recall(Positive)} . \tag{4.6}$$

$$Fmeasure(Negative) = 2 \cdot \frac{Precision(Negative) \cdot Recall(Negative)}{Precision(Negative) + Recall(Negative)} . \tag{4.7}$$

4.4.2 ROC Curve

The Receiver Operating Characteristics (ROC) curve is a graphical representation of classification performance. It is useful for evaluation because it provides a visualization of the performance of the classifiers [19]. The ROC curve depicts the true positive rate as a function of the false positive rate at different threshold settings. As mentioned above, the true positive rate (TPR) is the sensitivity or recall. The False positive rate (FPR), or false alarm rate, is the ratio of negatives incorrectly classified to the total number of negatives [19]. For reasons of completeness these rates are presented in the following equations:

$$TPR = \frac{TP}{TP + FN} . \tag{4.8}$$

$$FPR = \frac{FP}{FP + TN} . \tag{4.9}$$

The ROC space is a 2D space, where the y-axis represents true positive rate (TPR) while the x-axis the false positive rate (FPR). Each point in the space (x, y) represents a relative trade-off between TPR and FPR, i.e. trade-off between benefits and costs respectively. The diagonal line, as illustrated in Figure 4.2 , divides the ROC space into two areas. A confusion matrix, which generates one point in the ROC space, is characterized as good or bad prediction related to the diagonal. Specifically the closer a point is to the upper left corner, the better classification result it represents.

Most classifiers can produce a probability score for each instance, that represents the confidence with which this instance is assigned to the class. Ranking or Score classifiers (e.g. Naive Bayes) are based on continuous probability estimation, and produce such probability

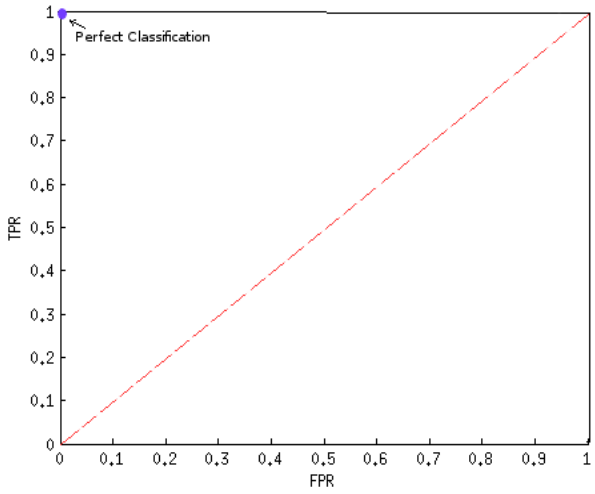

Figure 4.2: Diagonal line in the ROC space. The upper left point denotes perfect classification.

score directly. On the other hand, discrete classifiers (e.g. decision trees), that produce a discrete class decision corresponding to a single point in the ROC space, can be extended to convert a class decision to a numeric output. For example a decision tree can be transformed to return the proportion of training instances belonging to a particular class in a leaf, instead of the label of the majority class. Assume $count_p$ and $count_n$ the the number of instances in a single leaf with "match" and "mis-match" class respectively. The corresponding probability scores for that leaf are computed with the use of the following equations:

$$Score_p = \frac{count_p}{count_p + count_n} \ . \tag{4.10}$$

$$Score_n = \frac{count_n}{count_p + count_n} \ = \ 1 - Score_p \ . \tag{4.11}$$

Given a probability score, a classifier can decides the class of an instance, y imposing a threshold T on the score. Each threshold value produces a point in the ROC space [19]. Thus, a ROC curve is generated with the use of this threshold T, as a parameter in the range [0, 1]. In other words, we can say that ROC curve is a function of TPR versus FPR, parameterized by the varying threshold T.

In our approach we utilize the classifiers that are described in Section 3.2 . All of them, except decision trees, belong to the ranking/score classification category, producing a probability score for each instance. Decision tree classifiers are also made to return a numeric output, as explained above.

4.5 Results of the Trained Classifiers

In this subsection we present the results of the different classifiers that we used. To estimate the performance of our classifiers we utilize the k-fold cross validation technique. Due to the structure of our datasets, we split our sets of pairs into 7-folds, testing 14% of the database each time. As mentioned in Section 4.3, we evaluate the performance of he classifiers with the use of precision, recall, F-measure and ROC curves. The results that we present are the average estimates of the corresponding measures in each fold of the of cross validation.

In Tables 4.6 and 4.7 we can see the results produced by each classifier in each test case. P indicates the (positive) "match" class while N the (negative) "mis-match" class. Also in Tables 4.6 and 4.7 we notice that Precision, Recall and F-Measure for the positive class are low in relevance to the Baseline Average Classification. Due to the imbalanced data problem, some classifiers do not identify any "matching" pair in some user sets. Indicatively, in the *Twitter identification* case 93.10% of sets that were FN after the use of Decision Table classifier, did not contain any "matching" pair at all. The respective percentage for the Naive Bayes classifier is 91.90% and for KNN it is 92.70%. However the percentages for the negative ("mis-match") class are very high in every classifier, thus we achieve a very high accuracy as well.

Table 4.6: LinkedIn identification results for various classifiers.

Classifiers LinkedIn's identification	Euclidean Distance	Decision Tree	Naive Bayes	KNN (k=5)	NBTree	DTNB
Accuracy	90.16%	95.77%	93.41%	96.02%	95.77%	**95.95%**
Precision(P)	53.09%	82.28%	67.63%	83.73%	**86.96%**	85.77%
Recall (P)	**88.86%**	72.42%	61.68%	73.60%	66.67%	71.25%
F-measure (P)	66.47%	76.65%	64.33%	**77.95%**	74.93%	76.97%
Precision(N)	**98.54%**	97.04%	95.84%	97.12%	96.47%	96.94%
Recall (N)	90.47%	98.27%	96.79%	98.44%	**98.82%**	98.60%
F-measure (N)	94.33%	97.65%	96.31%	**97.77%**	97.62%	97.76%

In Figures 4.3 and 4.4 we can see the ROC curves for the *LinkedIn identification* and *Twitter identification* cases respectively. We notice that the DTNB classifier performs better in both cases (*LinkedIn identification* and *Twitter identification*), because its ROC curve is closer to the upper-left corner than the others.

To handle the imbalanced data problem we also tested another approach, where classifiers are used like rankers. Specifically, for each user set we choose as "matching" pair the one with the maximum probability. This probability, which is derived from the distribution of the positive class during training, denotes the likelihood membership of the instance in

Table 4.7: Twitter identification results for various classifiers.

Classifiers Twitter's identification	Euclidean Distance	Decision Tree	Naive Bayes	KNN (k=5)	NBTree	DTNB
Accuracy	91.99%	95.13%	95.67%	**96.02%**	95.49%	95.67%
Precision(P)	51.36%	74.74%	77.44%	**83.73%**	75.53%	77.92%
Recall (P)	**88.80%**	66.41%	68.34%	73.60%	66.41%	72.97%
F-measure (P)	65.08%	67.57%	70.88%	**77.95%**	69.31%	73.23%
Precision(N)	**98.92%**	97.04%	97.22%	97.12%	97.03%	97.58%
Recall (N)	92.32%	97.67%	98.11%	**98.44%**	98.04%	97.74%
F-measure (N)	95.51%	97.32%	97.64%	**97.77%**	97.51%	97.63%

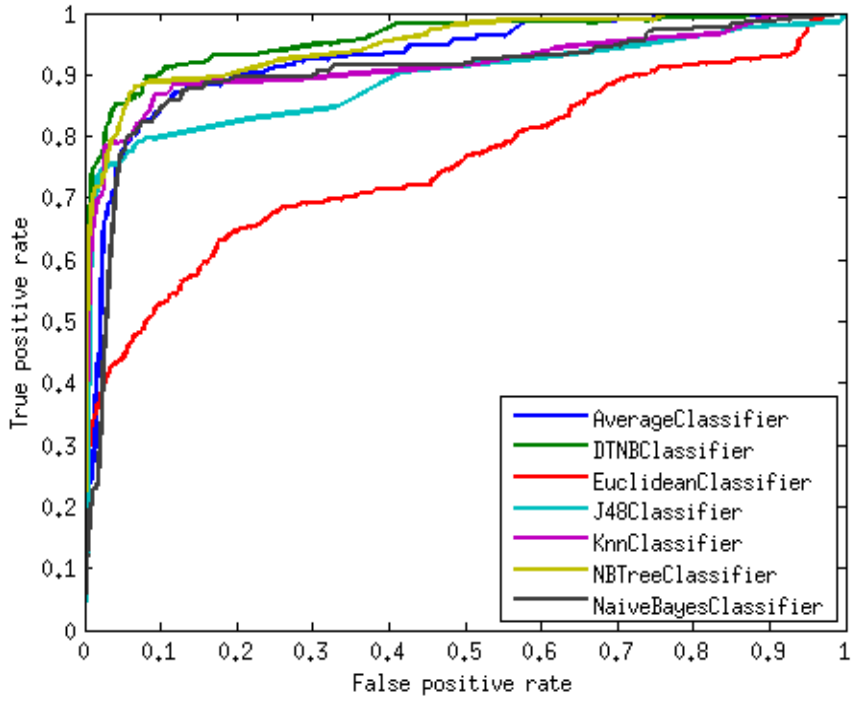

Figure 4.3: ROC curves of the classifiers for the *LinkedIn identification* task. The y-axis represents true positive rate (TPR) while the x-axis the false positive rate (FPR).

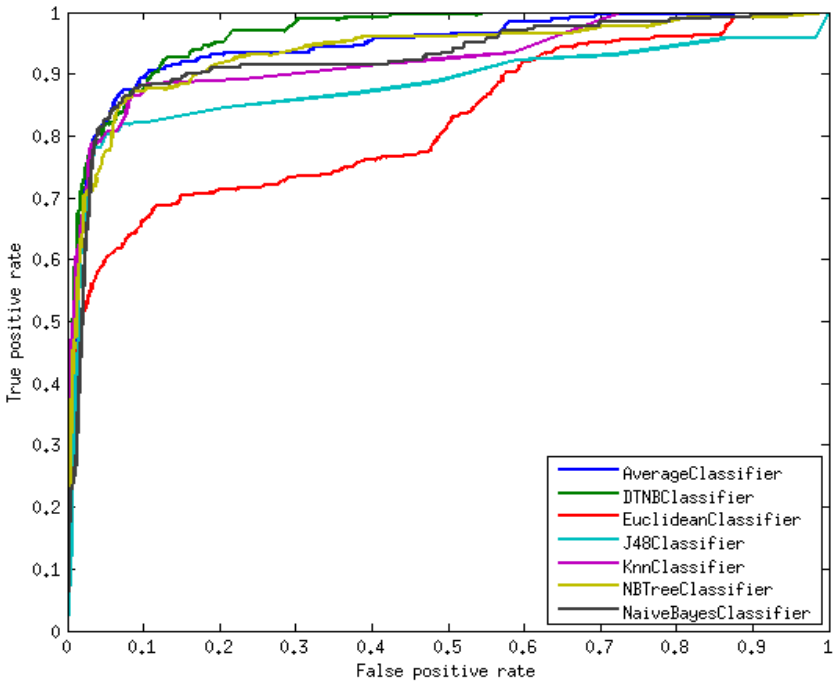

Figure 4.4: ROC curves of the classifiers for the *Twitter identification* task. The y-axis represents true positive rate (TPR) while the x-axis the false positive rate (FPR).

that class [20]. Tables 4.8 and 4.9 show the results in *LinkedIn identification* and *Twitter identification* for each classifier-ranker. The evaluation metrics represent only the "match" class, because the results for the negative ("mis-match") class are high due to imbalanced data problem, as we observed in the previous statement.

As shown in Tables 4.8 and 4.9, our approach performs well for detecting matching pairs, especially with the use of the DTNB classifier. Even the low proportion of ground-truth data, the results for precision and recall in match class are satisfactory, thus we achieve a high score in accuracy. In *LinkedIn identification* case, DTNB achieves approximately 95% in Precision, Recall and F-Measure in the positive ("match") class. Accuracy in DTNB

Table 4.8: LinkedIn identification results for various rankers.

Rankers LinkedIn's identification	Decision Tree	Naive Bayes	KNN (k=5)	NBTree	DTNB
Accuracy	97.87%	98.09%	98.40%	98.68%	**98.96%**
Precision	89.58%	90.35%	92.66%	92.28%	**94.98%**
Recall	88.96%	89.69%	91.99%	91.59%	**94.27%**
F-measure	89.27%	90.02%	92.33%	91.93%	**94.62%**

Table 4.9: Twitter identification results for various rankers.

Rankers Twitter's identification	Decision Tree	Naive Bayes	KNN (k=5)	NBTree	DTNB
Accuracy	97.93%	97.89%	98.57%	98.52%	**98.61%**
Precision	86.49%	86.10%	90.73%	**91.12%**	90.73%
Recall	86.49%	86.10%	90.73%	**91.12%**	90.73%
F-measure	86.49%	86.10%	90.73%	**91.12%**	90.73%

is very high (98.96%), due to high results in the "mis-match" class as well. In *Twitter identification*, if we take into account the results of Precision, Recall and F-Measure, we would notice that NBTree classifier performs better, achieving 91.12% in these measures. On the other hand, DTNB achieves a higher accuracy than NBTree (98.61%) and the results for Precision, Recall and F-Measure (90.73%) are close to the respective ones in NBTree. Taking into consideration the ROC curves as well, we can conclude that DTNB outperforms the other classifiers in both cases (*LinkedIn identification* and *Twitter identification*).

5. CONCLUSION AND FUTURE WORK

In this thesis, we studied user identification across two popular social networks, LinkedIn and Twitter. Our motivation was to verify the validity and trustworthiness of users in social networks. For the matching, we used different similarity measures for different pieces of professional information provided by the users in the two networks. The novelty of this work lies in the combination of different sets of features in order to produce the similarity measures. To achieve identification, we combined the similarity metrics using a variety of well-known supervised classification methods on the basis of similarity vectors. Specifically we tested our approach with the use of Baseline Average, Euclidean Distance, Decision Table, Naive Bayes, KNN Classifiers and hybrid techniques such as DTNB and NBTree. Also we handle missing values, as well as the imbalanced data problem by ranking the classification results and specifying as "matching" the pairs with the maximum probability. As shown in our experiments on the specific data set, using a hybrid classifier (DTNB) we can achieve a very high user identification performance.

A possible future extension of the presented work would be the handling of class imbalance with a more sophisticated approach. We could use ensemble filtering (e.g SMOTE) to over-sample the minority (match) class and under-sample the majority (mis-match) one [21]. Specifically, past experiments with SMOTE indicate the improvement of Naive Bayes and Decision Tree classifiers. Therefore, it would be interesting to use ensemble filtering in our classification strategy, and especially for NBTree because it combines Naive Bayes and Decision Tree classifiers. Another approach for handling the imbalanced data problem is to set higher weights to the positive (matching) instances during training [8]. The weight should be set properly, e.g. inversely proportional to the ratio of matches to mis-matches.

Moreover, we could enrich location information provided by the users with estimations of locations as mentioned by the users in tweets or in job descriptions, as [22] suggests. Chen et al. predicts the city-level location and points of interests of users by extracting content-based words in their tweets. We could modify this work by extracting specific fields in the LinkedIn network, in order to estimate professional location e.g. the headquarter of a company where a user works. Additionally, we would combine this with the work of Chen et al. [22] to produce a new similarity measure. This extension could increase the performance of our approach, especially in the journalism scenario, where many reporters are foreign correspondents.

Finally it would be interesting to study the potential contribution of our approach to

the difficult problem of identifying fake or compromised accounts in social networks [23]. The COMPA approach is addresses this problem, using event detection identifying sudden changes in user activity behavior. Event detection seem particularly suitable for the Twitter network, due to its nature. On the other hand, LinkedIn doesn't support events in public information of the users. Nevertheless, we could customize COMPA for LinkedIn by verifying the chronological sequence and content of jobs, education, skills and other professional characteristics of a user account.

A. APPENDIX

1.1 Used Toolkits

- Geonames: Geonames is a geographical database that provides geographical information through its web services and web ontology. Geonames' data consist of features, which are categorized into specific classes of toponyms such as region, lake, building etc. What makes Geonames attractive is that it integrates geographical data from many sources, providing a variety of geographical information (e.g. population, coordinates etc) in many languages. In addition, its web ontology is provided as a geospatial semantic taxonomy. Therefore, Geonames is helpful in our approach because we can receive useful geographical information with the use of its web services.

- SecondString: SecondString is an open-source Java library for named-entity matching. It provides many well-known string matching techniques, such as edit-distance metrics, fast heuristic string comparators, token-based distance metrics, and hybrid methods [13]. This toolkit computes a distance measure for each implemented metric, in order to specify the similarity of two entities. This library was helpful in the computation of our measures, which they are based on implemented metrics such as Levenshtein, Jaro-Winkler and SoftTFIDF.

- Simmetrics: Simmetrics is another open-source Java library that contains a variety of similarity metrics. This toolkit provides a plethora of techniques, ranking from simple ones, such as edit distance metrics, to more complicated algorithms such as Smith-Waterman 's. We utilized Simmetrics in the computation of the Affiliation-Education measure.

- Weka: The Machine Learning Group at the University of Waikato developed a software that includes several standard Machine Learning (ML) techniques. The Weka team provides a Java library, implementing many well-known ML algorithms, as well as many tools for data pre-processing, classification, regression, clustering, association rules, visualization and evaluation of algorithms and their results [20].

BIBLIOGRAPHY

[1] Reveal Project: Social Media Verification. `http://revealproject.eu/`. [Online; accessed 20-January-2015].

[2] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1):1–16, 2007.

[3] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, pages 73–78, 2003.

[4] Tereza Iofciu, Peter Fankhauser, Fabian Abel, and Kerstin Bischoff. Identifying users across social tagging systems. In *ICWSM*, 2011.

[5] Jan Vosecky, Dan Hong, and Vincent Y Shen. User identification across multiple social networks. In *Networked Digital Technologies, 2009. NDT'09. First International Conference on*, pages 360–365. IEEE, 2009.

[6] Anshu Malhotra, Luam Totti, Wagner Meira Jr, Ponnurangam Kumaraguru, and Virgilio Almeida. Studying user footprints in different online social networks. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 1065–1070. IEEE Computer Society, 2012.

[7] Oana Goga, Howard Lei, Sree Hari Krishnan Parthasarathi, Gerald Friedland, Robin Sommer, and Renata Teixeira. Exploiting innocuous activity for correlating users across sites. In *Proceedings of the 22nd international conference on World Wide Web*, pages 447–458. International World Wide Web Conferences Steering Committee, 2013.

[8] Oana Goga, Daniele Perito, Howard Lei, Renata Teixeira, and Robin Sommer. Large-scale correlation of accounts across social networks. Technical report, Technical report, 2013.

[9] Eric Roberts. Smith-Waterman Algorithm . `http://cs.stanford.edu/people/eroberts/courses/soco/projects/computers-and-the-hgp/smith_waterman.html`. [Online; accessed 26-April-2015].

[10] Erwan Moreau, François Yvon, and Olivier Cappé. Robust similarity measures for named entities matching. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 593–600. Association for Computational Linguistics, 2008.

[11] Geonames Ontology. `http://www.geonames.org/`. [Online; accessed 30-October-2014].

[12] Simmetrics Library. `https://github.com/Simmetrics/simmetrics`. [Online; accessed 2-November-2014].

[13] SecondString Library. `https://github.com/TeamCohen/secondstring`. [Online; accessed 2-November-2014].

[14] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Classification: basic concepts, decision trees, and model evaluation. *Introduction to data mining*, 1:145–205, 2006.

[15] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Classification: alternative techniques. *Introduction to data mining*, pages 207–223, 2005.

[16] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *KDD*, pages 202–207. Citeseer, 1996.

[17] Mark Hall and Eibe Frank. Combining naive bayes and decision tables. In *FLAIRS Conference*, volume 2118, pages 318–319, 2008.

[18] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[19] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine learning*, 31:1–38, 2004.

[20] Machine Learning Group at the University of Waikato. `http://www.cs.waikato.ac.nz/~ml/index.html`. [Online; accessed 15-March-2015].

[21] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, pages 321–357, 2002.

[22] Yan Chen, Jichang Zhao, Xia Hu, Xiaoming Zhang, Zhoujun Li, and Tat-Seng Chua. From interest to function: Location estimation in social media. In *AAAI*. Citeseer, 2013.

[23] Manuel Egele, Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Compa: Detecting compromised accounts on social networks. In *NDSS*, 2013.