Slovak University of Technology in Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

FIIT-5220-64357

Bc. Tomáš Jendek

Intelligent identity information processing

Diploma thesis

Degree course: Software Engineering
Study field: 9.2.5. Software Engineering
Department: Institute of informatics and software engineering, FIIT STU Bratislava
Supervisor: Ing. Radovan Semančík, PhD.
Education supervisor: Ing. Anna Považanová

May 2014

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Softvérové inžinierstvo

Autor: Bc. Tomáš Jendek
Diplomový projekt: Inteligentné spracovanie údajov o identitách
Vedenie diplomového projektu: Ing. Radovan Semančík, PhD.
Pedagogický vedúci: Ing. Anna Považanová
Máj, 2014

Správa identít je oblasť, ktorá sa zaoberá získavaním a spravovaním informácií o používateľoch v informačných systémoch, ich právach a prístupových účtoch. Práve informácie o prístupových účtoch a právach používateľov sú mnohokrát umiestnené v rôznych databázach a úložiskách údajov ktoré sú často heterogénne. V súčasnosti existujú systémy na správu identít, ktoré integrujú informácie z desiatok až stoviek podsystémov. Tieto podsystémy môžu v podnikovej sfére a v reálnom nasadení obsahovať tisícky záznamov o používateľských identitách. Čím viac podsystémov je integrovaných, tým je väčšia šanca vznikaní nekonzistentnosti.

Hlavným problémom integrácie je nedostatočná korelácia medzi záznamami o identitách. Existuje niekoľko prístupov ku tomuto problému, avšak ani jeden nie je dostatočne efektívny. Niektoré z týchto prístupov zahŕňajú korelačné výrazy a potvrdzovacie pravidlá, ktoré sú príliš jednoduché pre zložitejšie prípady nasadenia. Rovnako existuje mnoho prác o prepájaní záznamov a algoritmov pre hľadanie podobnosti znakov slov, avšak žiadny z prístupov neposkytuje verejne dostupné riešenie. Nedostatok automatických riešení má za následok manuálnu koreláciu identít, ktorá je síce najpresnejšia, avšak pre veľké organizácie časovo náročná.

V našej práci analyzujeme existujúce prístupy v oblasi správy identít, podobnosti záznamov o používateľoch a korelácie používateľov s využitím algoritmov strojového učenia. Predstavujeme návrh metódy na automatickú koreláciu údajov o identitách z rôznych systémov. Hlavný prínos našej metódy spočíva v automatizovaní procesu korelácie s využitím čo najmenej manuálnej práce. Naša metóda pomocou algoritmov na podobnosť reťazcov a algoritmov strojového učenia poskytuje možnosť spájať záznami o identitách z viacerých zdrojov. Implementáciu navrhnutej metódy realizujeme pomocou webovej aplikácie a overujeme pomocou experimentou s údajmi o zamestnancoch Slovenskej techniskej univerzity v Bratislave.

ANNOTATION

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Software Engineering

Author: Bc. Tomáš Jendek
Diploma project: Intelligent identity information processing
Supervisor: Ing. Radovan Semančík, PhD.
Educational supervisor: Ing. Anna Považanová
2014, May

Identity management systems manage various identity sources, integrate them and provide identities access to various heterogeneous systems. These identity sources often consist of records with various inconsistent attributes and thus integration can be difficult. Nowadays there are identity management solutions which integrate information from various stores, from tenths to hundreds of subsystems. Subsystems used in corporate sphere can include thousands of records with identity information. The more subsystems are integrated the more likely occurrence of inconsistent identity data is.

Main problem with integration is the proper correlation of identity records from various heterogeneous identity sources. Some correlating mechanisms were proposed, but neither of them is sufficient. For example correlation expressions and confirmation expressions are too simplistic to handle complicated scenarios. There are also many literature sources describing record linkage processes and string matching algorithms. However there is lack of open solutions for this problem. The lack of available automated solutions result in manual correlation, which is probably the safest way to correlate identities. But it is too time-consuming for larger identity management deployments.

In our work we analyze existing approaches in field of identity management systems, record matching, data deduplication and correlation of user records with using string similarity algorithms and machine learning approaches. We propose method for automatic correlation of identity records from various sources. Main asset of our work is automation of correlation process and saving manual work. Our method uses string similarity algorithms and machine learning algorithms for correlating identity records. Implementation is realized as web application and verification is done by experimenting with dataset from Slovak university of technology in Bratislava.

Table of Contents

1 Introduction

Identity management systems manage various identity sources, integrate them and provide identities access to various heterogeneous systems. These identity sources often consist of records with various inconsistent attributes and thus integration can be difficult. Nowadays there are identity management solutions which integrate information from various stores, from tenths to hundreds of subsystems. Subsystems used in corporate sphere can include thousands of records with identity information. The more subsystems are integrated the more likely occurrence of inconsistent identity data is.

Main problem with integration is the proper correlation of identity records from various heterogeneous identity sources. Some correlating mechanisms were proposed, but neither of them is sufficient. For example correlation expressions and confirmation expressions are too simplistic to handle complicated scenarios. There are also many literature sources describing record linkage processes and string matching algorithms. However there is lack of open solutions for this problem. The lack of available automated solutions result in manual correlation, which is probably the safest way to correlate identities. But it is too time-consuming for larger identity management deployments. Our aim is to design and develop effective method for user correlation with emphasis on the automation of matching process. This thesis is divided into sections. Section 2 describes principles in identity management especially identity stores, access management, provisioning, reconciliation. Section 3 describes identity correlation mechanisms like record linkage, string similarity algorithms etc. Section 4 presents analyzes of existing identity management solutions with focus on identity correlation. Section 5 presents correlation method proposal with data preparation process, normalization and detection methods. In section 6 we describe implementation of our correlation framework and dataset. In section 7 we describe experiments and results of our method.

We conclude with evaluation of our project and possibilities for further development.

2 Identity Management

Identity management manages identities in cyberspace. [5] It is also defined as combination of technologies and practices for representing and recognizing entities as digital entities in cyberspace. Identity management systems are not same for every organization, because of specific requirements of each organization. Main purpose of identity management system is integration of identity data, process and handle their life cycle including creation of new identities, modification and deletion.

[4] Person in digital world can be described as set of attributes which can be managed by technical means which is called digital identity (Figure 1). Digital identity uses personal data which can be stored and automatically processed by computer application. Term virtual identity is used as synonym to digital identity.

Identity in general is exclusive reception of life integrated into social group, which is bound to body and is constantly shaped by society around this identity. Identity is also any subset of attributes which identifies individual within set of individuals. "I" represents individual self as instance of liberty and initiative. "Me" represents social attributes which define human identity.

[21] Role is a set of connected actions taken by identity in specific social situations which is basically expected behavior. Technical description of identity is digital identity, which consists of attribute identifiers of individuality.

Partial identity is subset of person attributes both in real and digital world which represents person in specific situation or context. Usually person uses more than one partial identity e.g. for work, school or other activities. Partial identities contain information on person, which can be static (birthplace) or dynamic (phone number). Person may use different names – nicknames or pseudonyms.

User account stores all information about person in cyberspace. It can be also called user record, user identity or simply account. User account stores information about real world person for example surname or age. It also stores technical information in context of system, in which account was created for example account permissions or system resource settings.

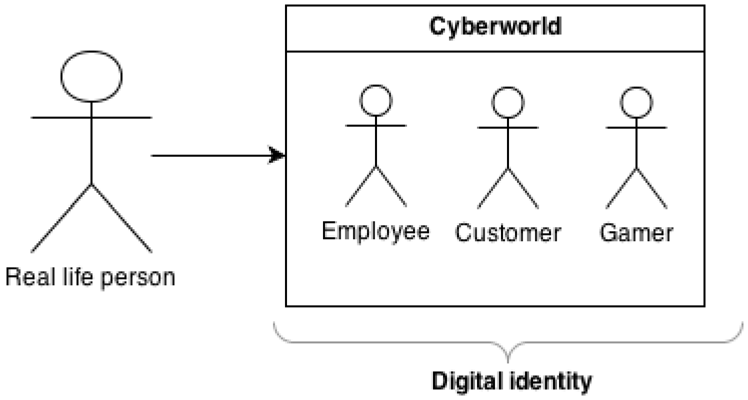


Figure 1. Person in cyberspace

Growth of Internet and distributed systems forced field of identity management to change from manual processes to fully automated processes. [6] There are three main environments with their specific problems within identity management: enterprise, internet and government. There are three areas of identity management technologies dealing with each environment:

- Enterprise identity management which takes place in enterprise environment and automate user management, authentication and authorization
- User-centric digital identity management take care about user's data in Internet environment
- Government digital identity management focus on data managed by government and has serious legal aspects of person's lives

Identity management is important for organizations which need to provide access to different subsystems for their employees or contractors. It is also need to manage life cycle of hiring new people by creating new accounts, modify them, or disabling accounts for fired employees. Every employee in organization has his own role, which need to be represented by technical means by providing access rights to resources. These roles and rights can change in time, so there is need to capture changes in system. Large organizations need to integrate numerous systems and subsystems and identity management systems reduce complexity and difficulty of integration process.

Identity management contains various technologies and we can define three main technology groups:

- Identity stores
- Access management
- Provisioning

2.0.1 Basic Principles in Identity Management

Anonymity and pseudonymity are core concepts preserved in identity management systems. [1] Anonymity is state of being unidentifiable or not uniquely characterized within anonymity set – set of subjects. Subject is acting entity i.e. human or computer. Subject anonymity can be enabled only if there is an appropriate set of subjects with same attributes. Anonymity ensures that user can use service without exposing his identity.

Pseudonymity uses pseudonyms as identifiers. Being pseudonymous is state of using a pseudonym. Pseudonyms are identifiers of subjects and he holder of pseudonym is subject which the pseudonym refers to. Pseudonyms are another kind of attributes widely used in IT systems because pseudonymity ensures that user can use resource or service without exposing his identity, but still can be able to use it. Digital pseudonyms are strings, which has to be meaningful in certain context. It also must be unique as an identifier and must be able to authenticate the holder's actions.

2.1 Identity Stores

Identity stores hold information about user's accounts and are often shared with other application within organization trough network. Identity stores use various technologies, especially Directory Services which provide storing user and accounts in tree structure for example LDAP (Lightweight Directory Access Protocol). Active directory is directory service which provides authentication and authorization users and computers within Windows domain type networks.

Directory services often use [27] LDAP protocol, which is application protocol for accessing distributed directory information services over network. Directory information services can provide structured records such as organizational email directory. LDAP protocol is very popular because of his scalability.

Identity store can be part of one application, or it can be shared with more than one application (Figure 2).

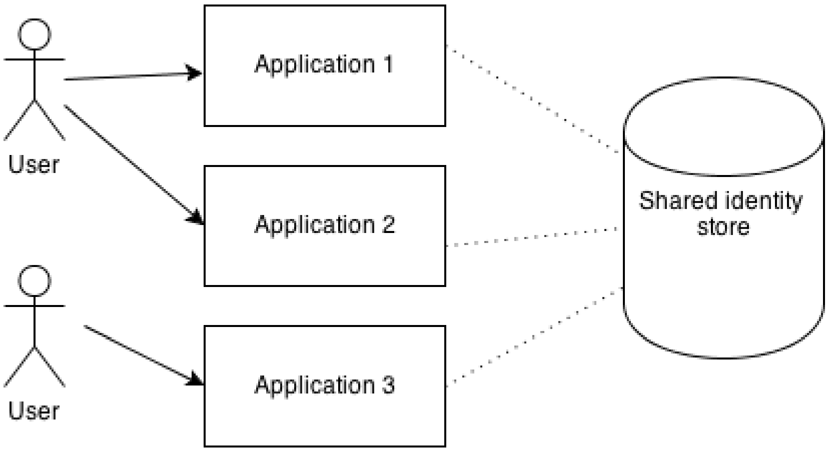


Figure 2. Shared identity store

2.2 Access Management

Access management manages user authentication and authorization and unifies security processes. In many systems the importance of resource security is paramount, especially in identity management which deals with sensitive personal data.

Authentication is part of access management which provides verification of users. It is the process of establishing who a person is and creating trust relationship between system and consumer of services. There are many options for authentication for example using matching process between public identifier (user name) and private identifier (password) or digital certificates.

Authorization takes place when user is successfully authenticated and need access to system resources. Authenticated identity has certain set of permissions and authorization deals with determining which permissions will be granted to identity.

[8] Single sign-on (SSO) is concept that provides user to authenticate once and gain access to all systems without logging again. Single sign-on concept is part of access management. User has to remember only one password to get access to many different systems and resources. [7] There are four types of SSO:

- Enterprise SSO connects systems within same enterprise
- Multi domain SSO connect multiple systems across multiple enterprises
- Web SSO connects applications and services across the web
- Federated SSO connect systems based on federated identities by combining identity attributes from multiple IDM systems

There is a variety of architectures which can be used for SSO implementation:

- Broker-based SSO
- Agent-based SSO
- Token-based SSO

[10] In Broker-based SSO solution there is one server for central authentication and broker gives electronic identity that can be used for requesting access to various systems. [28] Kerberos is an authentication protocol for TCP/IP networks and it is basic model for a broker-based Single Sign-on architecture. It uses trusted Kerberos server which is actually broker. Kerberos server centrally authenticates users and give them electronic identity based on the credentials given. After the user was authenticated on server, he gets ticket for different services and applications.

Agent-based SSO solution is based on agent program placed on server side acting as interpreter between authentication system and authentication method used by client. The SSH is an example of agent-based solution.

[9] Token-based SSO uses physical token that generates time dependent one-time passwords for user authentication. When user logs on system, a centralized authentication server will authenticate user and generates user token including session key and time stamp. User can use token to access various application servers. These servers send users request to centralized authentication server for token validation. SecurID is one of implementations of this token and is based on synchronized clock on hardware token and network server. Generated password is accepted only within certain time window.

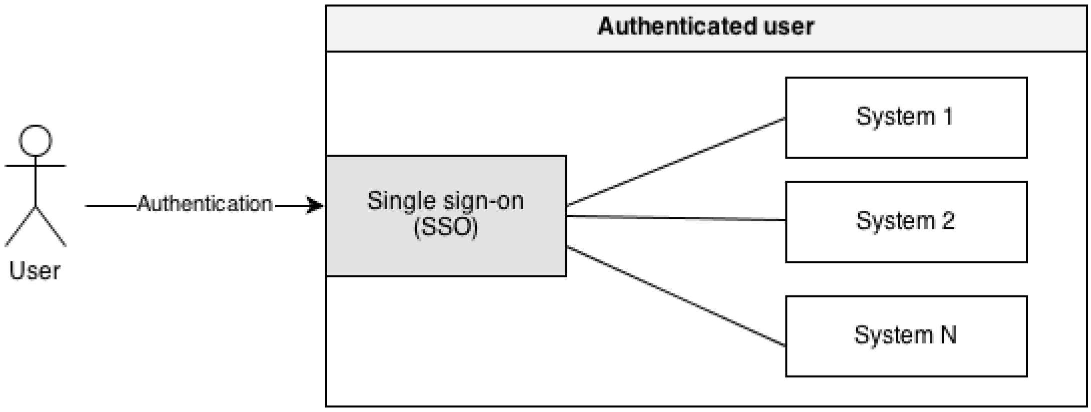


Figure 3. Single sign-on schema

2.3 Provisioning

Provisioning part of identity management systems take care of managing and integrating many identity stores. Provisioning systems synchronizes various data and models from many sources by replicating changes to the different resources. Provisioning includes complex rules and expressions mechanisms to match models of connected systems and stores.

For example hiring new employee starts with creating record about person in human resource system. Provisioning system then automatically detects new record and assign role to user. Based on role provisioning system creates accounts in the external systems.

Another example can be when work position of an employee changes and new role must be activated. Provisioning system detects change made in the human resource system , changes the role and creates new accounts.

[14] Identity management systems provide alternatives to provision resources to authorized users on request-based, role-based and hybrid approach. In request-based approach, users request access to special applications and resources with certain privilege levels within system. Requests are validated by workflow driven approvals. Administrators are alerted to new or unused accounts and have an option to activate, modify or delete such accounts.

Role-based provisioning approach automates process of granting access to resources. Users are assigned to roles and get specific set of accounts and access rights based on their role. User can be removed from role and entire set of corresponding accounts and rights are removed.

Hybrid approach combines request and role-based approach. Automated role-based assignment access rights and accounts can be enriched by providing options to manage accounts manually.

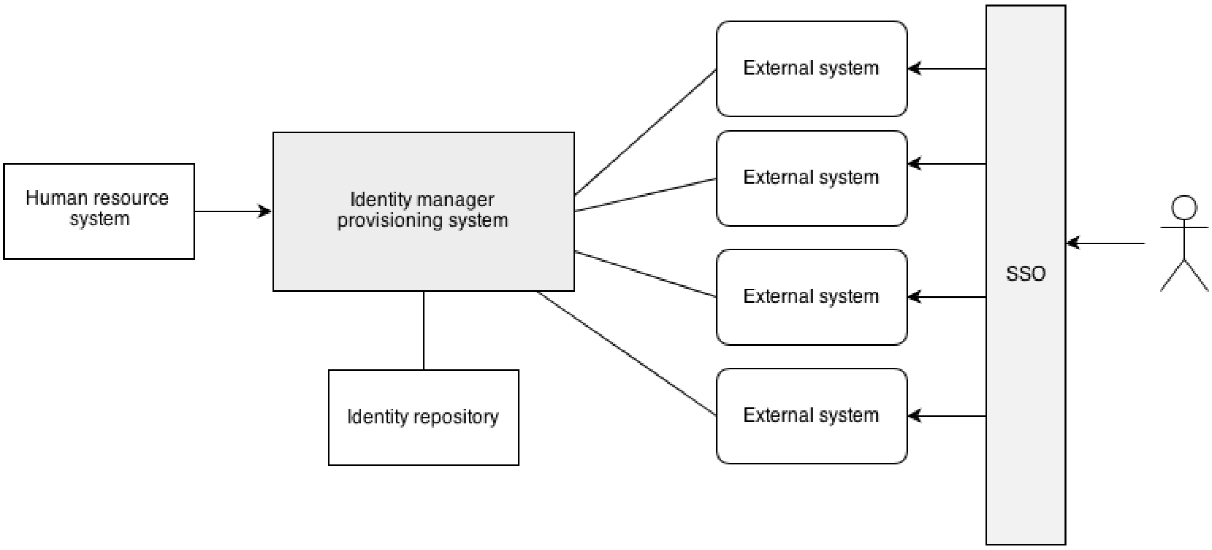


Figure 4. Identity management system

2.4 Role-Based Access Control

Large organizations has to deal with unauthorized access to organizational resources, applications or external systems. Role-based access control (RBAC) is one of the most known access control standards. It simplifies access control policies by grouping users in roles, which are ordered in a role hierarchy. [12] There are overlapping responsibilities and privileges that users can have within organization and users with different roles may need to do common operations. A role hierarchy defines roles that can contain other roles which mean that one role can use operations from another role. Role is abstraction that contains set of responsibilities with corresponding allowed operations. Privileges are assigned to role which means that certain role has predefined set of operations within system. As shown on figure 5, user cardiologist has role Cardiologist which contains privileges of doctor role and intern role.

[11] RBAC abstraction provides security administration at business enterprise level rather than at the user identity level. Functional roles in organizations are captured as role with defined permissions and role, or set of roles are assigned to user.

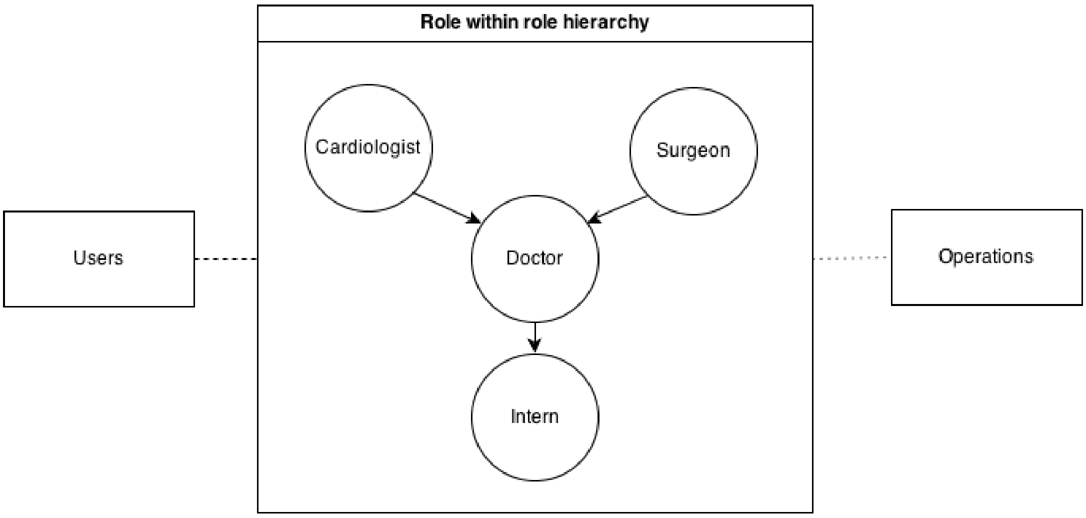


Figure 5. Role hierarchy

2.5 *Reconciliation*

Reconciliation is process of synchronization user account from various resources and purpose is to create user-centric identity system that holds single profile with links to accounts in other systems. For determination of an ownership reconciliation compares account information between information from user accounts. [22] There are multiple options for reconciliation for example automatic matching accounts with consistent unique identifiers, matching other attributes or using mapping tables if they are available in organizations. Reconciliation process is based on following steps:

- Reading relevant information from the source system
- Match the information from the source system to existing information in the identity store using a correlation function (correlation rule)
- Read relevant information from identity store
- Compare the information retrieved from the source system with the information in the identity store and calculate differences
- Perform defined actions based on calculated differences

Information represents user attributes and permissions or roles. Actions to be performed are for example modification of user when account is linked, delete or unlink account, resolve collision etc.

3 *Identity Correlation*

Identity can operate under various accounts in different systems with different names (Jhsmith, James_smith, JS54), addresses etc. as shown on figure 6. Identity management systems need to keep user accounts linked, so that multiple user accounts from various systems can be easily changed. For example if the surname of employee changes after marriage, identity management system changes this name in all subsystem accounts. Or in case of firing employee, all accounts must be disabled or destroyed which is called de-provisioning. Identity correlation is process of reconciliation and validation of multiple user accounts which are linked by individual ownership. Identity matching is usually done by comparing user attributes using expression languages. Identity correlation is part of account linking process. Links are usually created automatically, but there is need to manually verify results of linking. Manual linking does not scale and is not efficient.

Identity management systems provide integration of various identity stores containing heterogeneous data. Usually there is no single authoritative source, so integration can be very difficult. Main goal of integration is to map identities or user accounts from various sources using for example rule engines, or expression languages.
Integration is used by synchronization mechanisms which checks correctness of user account state - consistency.

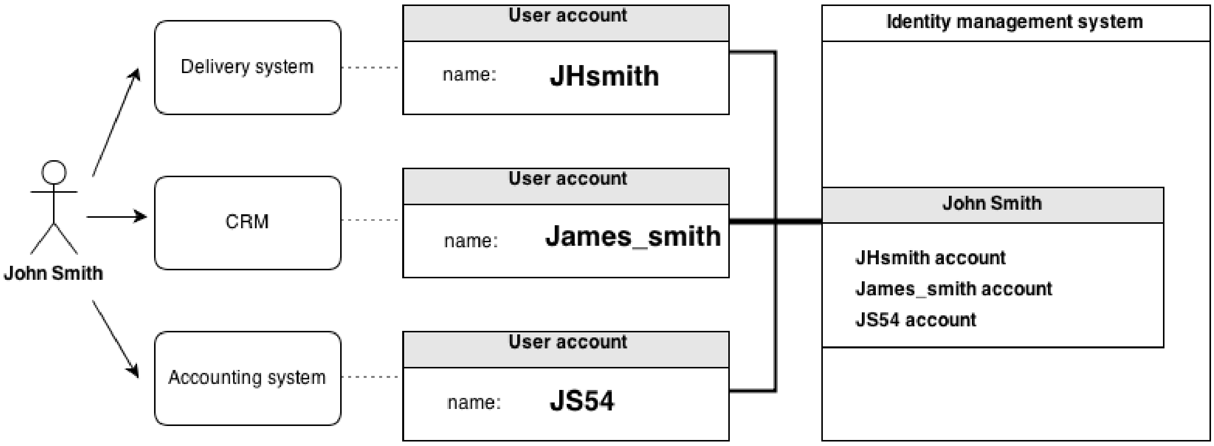


Figure 6. Identity correlation

Existing provisioning systems usually use simple correlation expressions to match the identities as shown on figure 7. These expressions get information from the account and build search query for finding owner of an account. Correlation expression is usually parametric search query, i.e. a search query with some parts determined by a dynamic expression. When correlation expressions match two or more accounts, confirmation expressions take place. Confirmation expression provides comparison of potential users accounts.

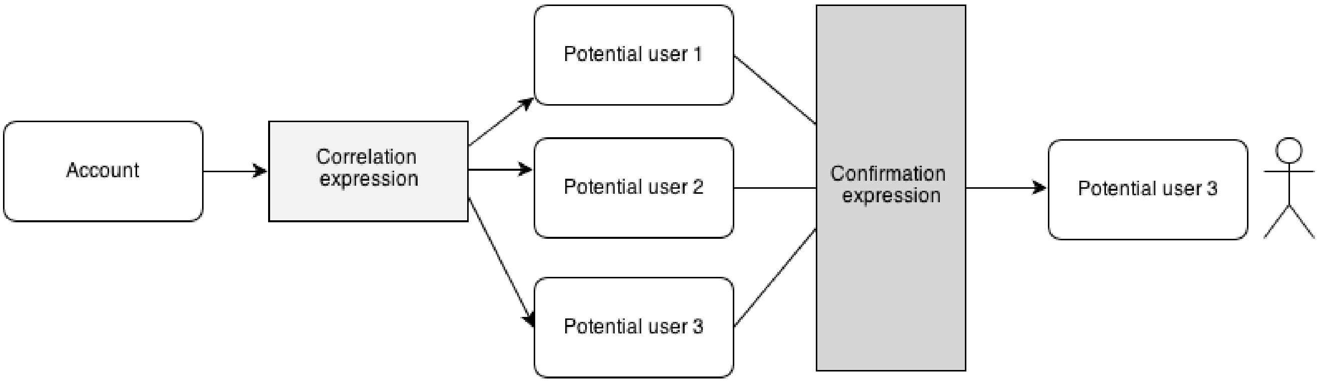


Figure 7. Correlation and confirmation expression schema

3.1 Record Linkage

Organizations often need to identify and match records within large databases. Especially in systems which are dealing with personal data integration.

User account records in databases consist of attributes for example name, address, date of birth etc. These attributes often contain typographical errors (misspelling, missing letters, incomplete words, incorrect or missing punctuation, abbreviations and fused or split words), data representation across sources may differ or change in time, which make duplicate identification very difficult.

[15] Record linkage is methodology of matching corresponding records from two or more sources or finding duplicates in files. Identity management deals with situations where several account records from various identity sources may refer to the same real world entity while not being syntactically equivalent. Set of records which refer to the same real world entity can be in general interpreted in two ways. First is to take one of the records as correct and the other records as duplicates containing errors. There is need to clean error duplicates. Other way is to merge matching records as partial sources of information to create one complete record. This is common with identity management systems, where is need to map records but not necessarily change them.

Record linkage involves bringing potential matches together for comparison including duplicate detection. It also involves comparison of potential record pairs whether they belong to the same real world entity. There are two approaches:

- Probabilistic linkage
- Deterministic linkage

Deterministic linkage is also called exact due to exact one to one matching character within linkage variables with one high quality identifier. Probabilistic linkage uses combination of the partial identifiers for example first name, email or address to compute weights for each potential match based on probabilities.

[17] Duplicate detection in record linkage is straightforward method for revealing exactly same records – real world entities. Records are sorted in a table and then

neighboring tuples are checked. This approach could be also used to detect approximate duplicates. Sorting is based on application-specific key for example first name and last name so that likely records appear near each other. There are duplicate detection algorithms using sliding window of fixed size for sorted records. The size of window is *W* and *i* is record, *i* is compared with record *i-W+1*trough *i-1 if i* > W, and otherwise with records 1 trough *i*-1. Repetitions and combining results of sliding window matching with small window size lead to better results as one repetition with large size of window.

[16] Traditional approaches to duplicate detection are on approximate string matching criteria. It can be enriched with domain specific rules. Recently there have been new adaptive approaches which use attributes and labeled data. Persons similarity is enriched with additional attributes for instance if two accounts refers to same location, or same workgroup, it is highly probable that these accounts belongs to same person.

Record linkage matching results are dependent on attribute value having errors and inconsistencies. Different attributes need different metrics when comparing values.

[20] Record linkage process consists of five steps (figure 8):

- **Cleaning and standardization** deals with data errors and inconsistencies by converting attributes to same format, adding derived variables
- **Indexing / blocking** generates candidate record pairs
- **Comparison** results are weight vectors that contain numerical similarity values
- **Classification** is based on weight vectors and results are of type *matches, non matches, possible matches*
- **Evaluation** of quality of generated matches and non matches. Often manual review is needed to decide final linkage

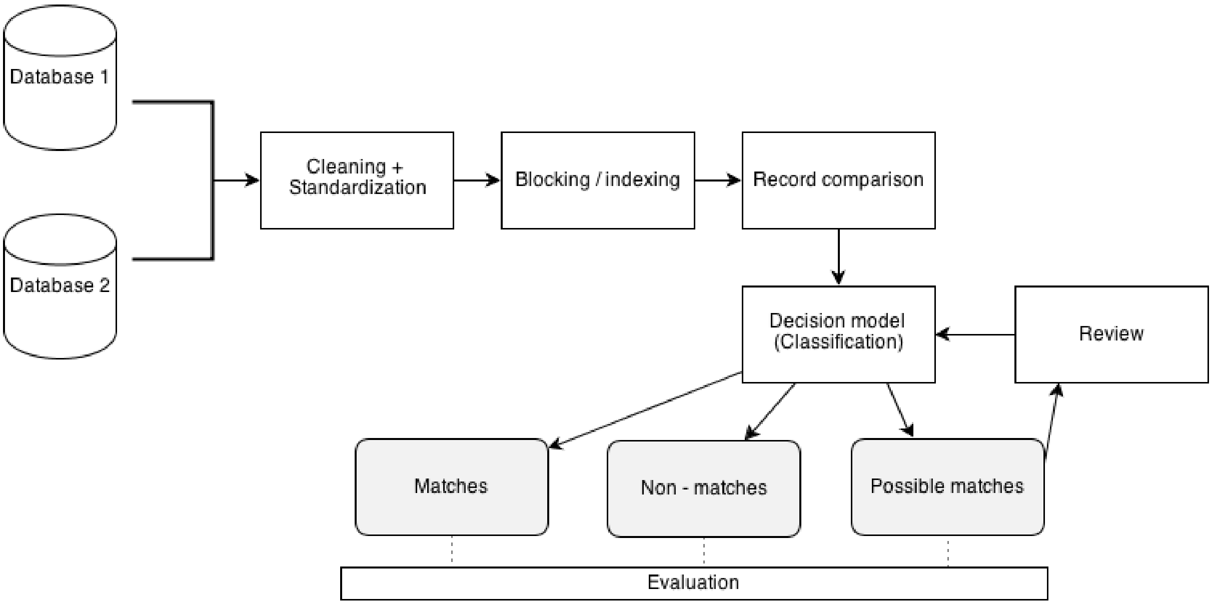


Figure 8. Record linkage process overview

3.2 Typical Errors in Matching Variables

[23] Sources of errors in matching personal records are for example misspellings, using phonetic name, use of synonyms or nicknames, lack of initials, compound names etc. Most common errors are:

- **Present surname** can change due to marriage or divorce. There are also compound surnames where birth surname and marriage surname are mixed.
- **First forename** brings errors with variations of forenames due to transcription or modifying forenames caused by fashion trends (popular persons). People often use nicknames instead of forename.
- **Address** can change during person's life quite often. There are also problems with using mailing addresses and physical addresses.
- **Date of birth** is variable used to verify age of person, but there are also problems in format of date (European, US).
- **Swapping names and surnames** is frequent error caused by transcriptions.
- **Titles in name variable** are for example marital status, academic title, academic degree, church, family order, concatenated names etc. Titles cause problems with name matching due to difficulties with parsing name to forename and surname.

3.3 String Normalization

Preprocessing of record information or attributes is most suitable before using string comparison methods. Normalization can make string comparison easier, sometimes even basic string matching is enough. Normalization can be considered at cleaning and standardization level if it does not spoil information value in attribute. Basic normalization methods are:

- Transformation of all characters to lowercase
- Removing whitespaces before and after string
- Removing punctuation characters or replacing them

Special normalization techniques can be applied in addition to type of information or attributes in identity records. For example if attribute is name of person, then we can extract academic or other titles into new attribute and then we can user or process these information later. Another example is address attribute, which can contain compound address information (part of the town) and this can be extracted to new attribute and used later.

3.4 String Comparison Methods

There are several methods for record linkage i.e. entity name clustering and matching, edit distance, vector space cosine similarity. Some recent works combine multiple standard methods and metrics.

Similarity estimation can vary depending on the domain. For string similarity improvement there is need for adapting string similarity metrics for each field corresponding to the particular domain.

[3] Methods for string similarity can be divided into two groups:

- Character based techniques
- Vector space based techniques

Character based techniques rely on character edit operations such as deletions, substitutions, insertions, comparison of subsequences. Such technique is Levenshtein distance which is defined as minimum number of insertions, deletions or substitutions necessary to transform one string into another string. Character based techniques work well for estimating distance between strings with typographical errors or abbreviations, but these metrics are computationally expensive and also less accurate for larger strings.

Vector space techniques deals with this problem better, because such techniques are based on viewing strings as bags of tokens. The order of tokens is unimportant. Strings in tokens are represented as sparse n-dimensional vectors of real numbers where every component corresponds to a token present in string. TF-IDF is probably most known method and useful for larger strings and text documents.

Records can be composed of multiple attributes and distance between these records must combine similarity estimates for each attribute. Each attribute can have different informative value and thus is necessary to weight attributes properly.

In [2] the authors propose object identification system based on domain independent string transformations to compare objects shared attributes. They use candidate generator which use set of domain independent transformation to judge similarity between objects. Candidate generator produces an initial set of candidates. The authors propose unary transformations which are used to determine candidates:

- *Equality* for testing if a token contains same character in the same order
- *Stemming* converts a token into its stem or root
- *Soundex* converts token into a soundex code. Tokens that sounds similar have same code
- *Abbreviation* looks up token and replaces with abbreviation

N-ary transformations:

- *Initial* computes if one token is equal to the first character of other token
- *Prefix* determines if one token is equal to a continuous subset of the other starting at first character

- *Suffix* determines if one token is equal to a continuous subset of the other starting at last character
- *Substring* computes if one token is equal to a continuous subset of the other but not include first or last character
- *Acronym* computes if all characters of one token are initial letters of all tokens from other objects
- *Drop* determines if a token does not match any other token

Tokenization is process of lowercasing all characters in text and removing punctuation characters. Transformations are used after tokenization.

Matching variables in records contain string values. There are several solutions for string comparison for example:

Levenshtein distance (edit distance), [29] which is defined as smallest number of insertions, deletions or substitutions of characters needed to change one string to another. Levenshtein distance can be modified to provide different edit costs – weights for edit operations in special situations depending on domain in which Levenshtein method is used. It can be effectively used in some situations i.e. "I" and "L" can be mechanically scanned as same letter and we need to give lower edit cost to operations witch these characters.

Damerau-Levenshtein distance [30] is variation of Levenshtein distance. Transposition is new operation and it costs just one edit instead of deletion and insertion. It is often used when error rate in string is low (misspellings).

Brute force string comparison is simplest algorithm to use. Algorithm try to match all possible pattern positions in string and verifies that pattern at exactly same position. If one string contains *x* characters and second string contains *y* characters, then in worst case there are *x.y* comparisons.

Knuth-Morris-Pratt (KMP) [31] algorithm is faster than brute force algorithm because of using sliding window over the strings in text. It does not try all positions as brute force, but it reuses information from previous check.

Boyer-Moore algorithm works similar to KMP, but check inside the window can proceed backwards and forwards.

Bag distance is cheap approximation to edit distance. A bag is defined as a multiset of the characters in a string (for example, multiset ms('peter') = {'e', 'e', 'p', 'r', 't'},
and the bag distance between two strings is calculated as
distbag(s1, s2) = max(|x − y|,|y − x|), with x = ms(s1), y = ms(s2) and |·| denoting the number of elements in a multiset.

Smith-Waterman is algorithm suitable for names with initials and compound names. It is based on a dynamic programming approach similar to edit distance, but allows gaps
as well as character specific match scores.

Longest common sub-string (LCS) repeatedly finds and removes the longest common sub-string in the two strings compared, up to a minimum lengths. This algorithm is recommended for compound names and names where first name and surname are swapped.

Q-grams are sometimes called n-grams are substrings of length (q,n). If q = 2, then name “Thomas” is split to bigrams “Th”, “om”, “as”. Similarity is calculated between splited n-grams so that similarity counts grams which are common.

Positional q-grams is extension to q-grams and add positional information and match only grams within certain distance.

The Jaro distance algorithm is used for name matching in data linkage systems. It counts for insertions, deletions and transpositions. The number of common characters and number of transpositions are used in this algorithm.

Results in [18] shows that for names parsed into separate fields, Jaro algorithm performs well with given and surnames. They also recommend knowing data, types of names and separators before choosing matching algorithm.

In paper [19] the authors present novel person name matching model. They formalize name variations in English language, introduce name transformation paths. Subsequently supervised techniques are used to learn a similarity function and decision rules. Transformation paths are weighted to give reasonable results and similarity function counts with these weights to improve estimations. They use support vector machine to (SVM) to learn a decision rule.

3.5 Machine Learning Data Correlation

Machine learning provides algorithms that automatically improve their performance based on gaining experience. Generating predictions is core functionality of machine learning algorithms. Algorithm can learn - improve predictions based on example data inputs. Recent research shows that there is no generic learning approach for all cases and in fact, different algorithms can produce similar results. The nature of data used to characterize task influence success of a learning algorithm. Data must be statistically regular and that is condition for learning algorithm to provide reasonable results.

Machine learning discovers regularities and classifies data, which must be preprocessed (removing redundant or irrelevant data) in order to provide less time consuming computations. Classification task in machine learning is based on generalization from the training objects to provide new object to be identified as belonging to one of predefined classes. [24] Predefined classes with specific example objects (training objects) are labeled and this is called supervised learning. In unsupervised learning, there are no predefined objects labeled with appropriate classes. Supervised learning is dependent on quality of data provided for training. Authors used Levenshtein distance with affine gaps, where affine gaps are sequence of non-matching characters. This method provides better results with abbreviations and can be modified using parameters for penalization certain affine gaps in special cases. For example various characters has different meanings within attributes – numbers in address are more important than number in person's name. Authors use support vector machine (SVM) for classification.

Case-based learning algorithms (CBL) is presented in [25], where authors present CBL algorithms as good choice for supervised learning tasks and are describing framework for CBL algorithms. They focus on learning issues and do not perform case adaption and smart indexing schemes. There are various CBL algorithms, for example Protos and MBRtalk, which were applied to a large range of tasks with considerable success. Overall experience shows that algorithm which work with one application, does not ensure that it will work for other.

3.6 K nearest neighbor

[32] K nearest neighbor algorithm is one of the simplest decision procedure. It classifies samples in addition to the category of nearest neighbors. K nearest neighbour algorithm assigns to a test pattern the class label of its k-closest neighbour(s) by using majority vote. The value of k is the most important, beacause the right value can improve accuracy. There are modifications of this algorithm for example modified k nearest neighbor algorithm, where nearest neighbors are weighted according to their distance from test node. Instead of using majority vote for classification, a weighted majority rule is applied. As seen on figure 9, testing point (green) will be classified as red triangle if k = 1, 2, 3. If k = 5, class is blue square.

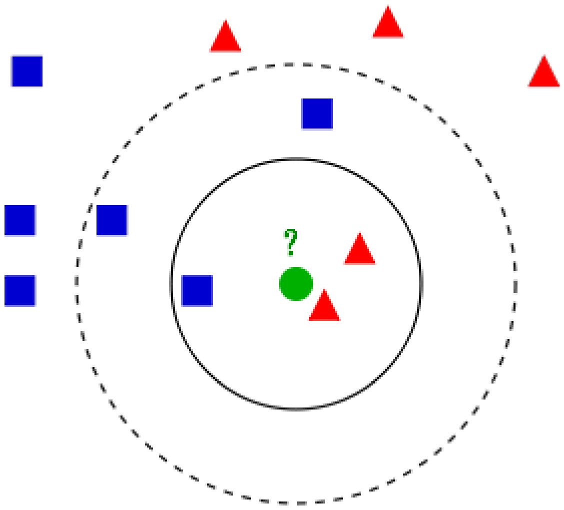


Figure 9. K nearest neighbor example

3.7 Support vector machines SVM

The support vector machine is binary classifier. It creates a decision boundary in multi-dimensional space by using sub-set of training set vectors. [32] The elements of sub set are support vectors. Support vectors are geometrically those training patterns, that are closest to the decision boundary. For determination of classes, the linear discriminant functions can be used. In general decision boundary is obtained as hyper-plane for separation of testing nodes. An SVM model is representation of the example nodes as points in space which are mapped so that the examples of different categories are divided by a clear gap which is as wide as possible. New examples will be mapped into the same space and predicted to belong to a category in order to position in gaps. SVM can maximize the margin around the separating hyper-plane.

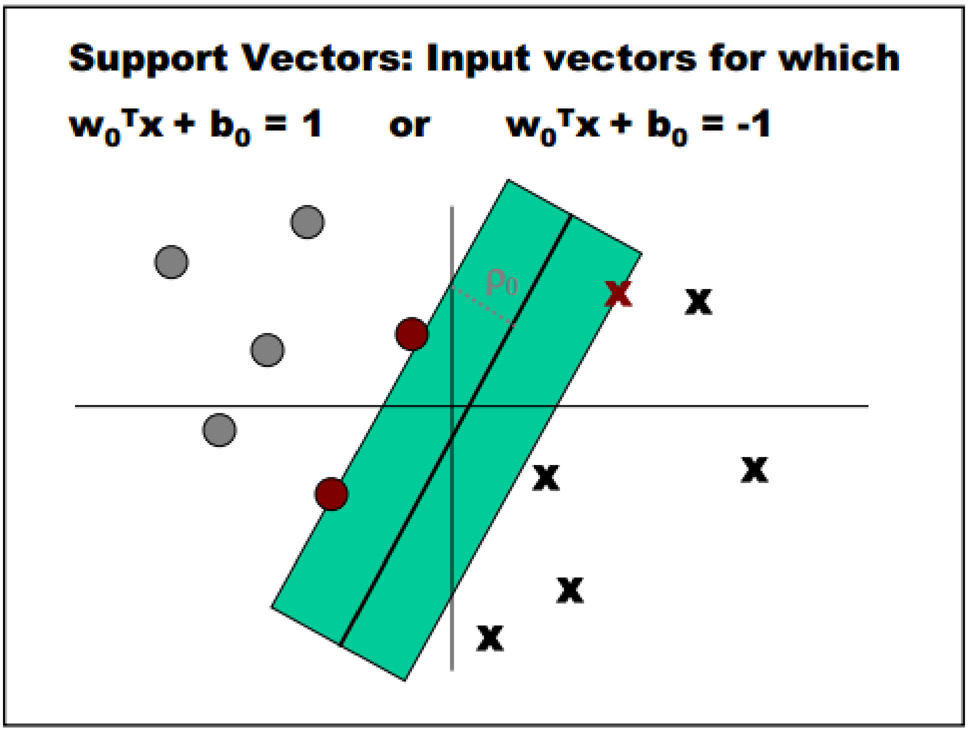

Support Vectors: Input vectors for which
$w_0^T x + b_0 = 1$ or $w_0^T x + b_0 = -1$
ρ_0

Figure 10. SVM hyper-plane example

Support vector machine classifier is used in text categorization, images recognition, medical science and hand written characters recognition.

3.8 Neural networks

[32] Artificial neural network was based on observing how human brain works. The output of neural network depends on inputs and weights in the network. The training of neural network consists of making the network give the correct output for every training input. Every link in the network gets random weight and if the output is correct, weights are not changed. Otherwise new random weights are created. This procedure is repeating until all inputs have correct output. Neural network consist of artificial neurons that are modeled as neurons in human brain. The input in the neuron is weighted and summed up. If aggregation exceeds a threshold, neuron outputs signal. Neural network models are

mathematical models that define functions. Network functions are made of other predefined functions such as hyperbolic tangent. Neural networks are used in robotics, data processing, clustering etc.

3.9 Decision trees

[32] Decision trees are commonly used data structures in pattern classification because of high transparency. A decision tree is a tree where non-leaf nodes are associated with a decision and the leaf nodes are associated with class label. Each internal node test one or more attribute values and links to another node. Decision trees are good for choosing between several courses of action.

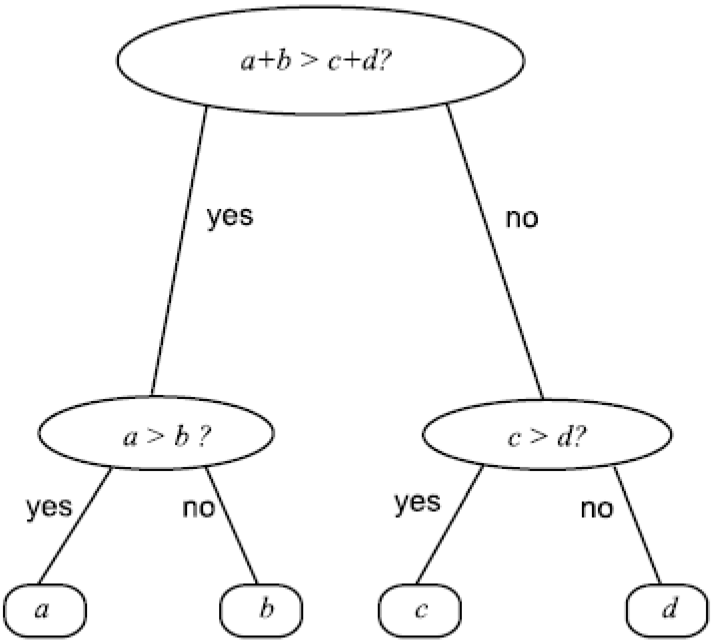


Figure 11. Simple decision tree example

For patterns classification using decision trees, the nodes represents status of the problem after making decision. The leaf nodes are labels of the classification rule based on the path from the root node to leaf node. In decision trees both numerical and categorical features can be used. The tree can be binary or non-binary, so that we can decide between many options. The rules are simple and easy to understand. Cons of decision trees are time difficulty for construction of the tree. There are many construction algorithms for example ID3, C4.5 etc.

4 Account Correlation in Identity Management Systems

There are existing solutions in field of account correlation in identity management systems. There are open-source projects such as OpenIAM or OpenIDM and commercial solutions like Oracle Identity Manager. We present short overview on these systems with focus on provisioning and reconciliation part of these systems which partly deals with account correlation.

4.1 Oracle Identity Manager

Oracle Identity Manager is an enterprise identity management system that automatically manages user's access privileges within enterprise resources. System provides secure access management for applications, data, web services and cloud-based services. It also provides single sign-on, authorization, mobile and social sign-on etc. Oracle Identity manager provides identity governance user self service, which simplifies account administration. Identity governance provides user registration, access requesting, role lifecycle management, provisioning, access certification etc. Oracle Identity Manager uses correlation and confirmation rules for finding user account owners and mapping accounts. Correlation rules consist of object attributes – account representation used for attribute based search and list of attribute conditions which determine list of potentially matching users.

After deploying Oracle Identity Manager infrastructure definitions of security polices take place which determine what data users or applications can access. These polices are stored in access control lists in Oracle Internet Directory. User identities are provisioned in Oracle Internet Directory. Identities come from multiple sources for example human resources applications or user administration tools. These identities, groups and roles are synchronized with other directories. User identities, groups and roles are associated trough provisioning process which can be performed manually or automatically trough provisioning integration.

4.2 Open IAM

Open IAM is open source identity management system based on Service Oriented Architecture. It is one of the oldest open source provisioning systems. Services like identity service or audit service are exposed to users and administrators by Enterprise Service Bus (ESB). Open IAM provides identity management functionality like identity life cycle managing, provisioning, synchronization etc. Provisioning manages accounts based on rules or roles. Audit logging and reports are part of provisioning module. Synchronization functionality allows to synchronize information from several authoritative sources. OpenIAM synchronization is based on:

- Events which allows real time synchronization by sending message to Identity Manager Bus and then synchronization starts
- Scheduled intervals are precisely configured time intervals in which synchronization can be done.

Reconciliation detects changes in managed systems for example if Active Directory make change, reconciliation mechanism based on rules take place and synchronize this change with OpenIAM directory.

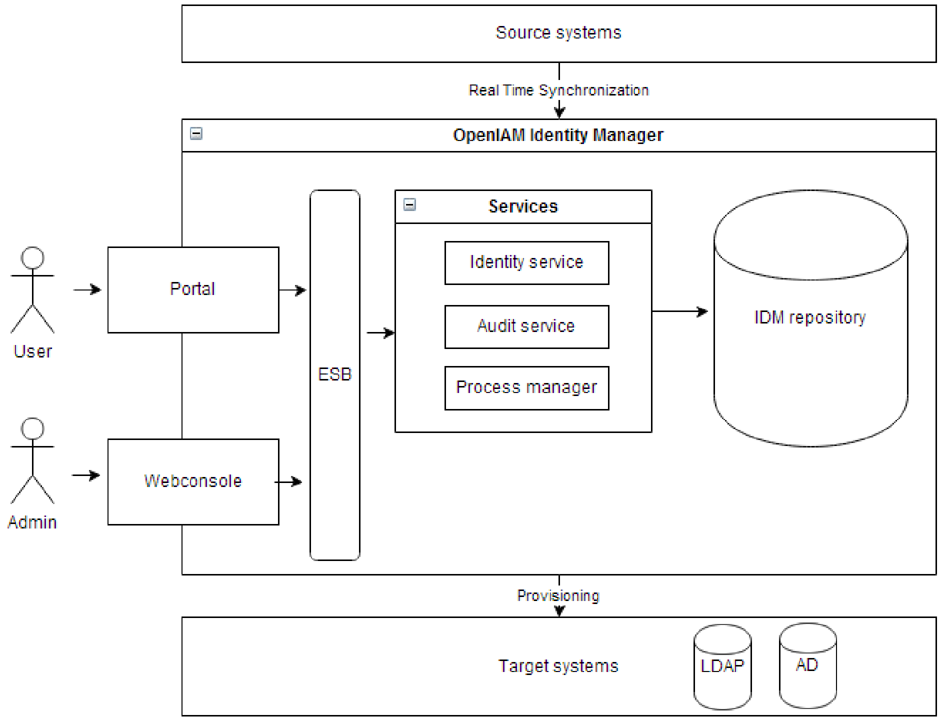


Figure 12. OpenIAM architecture overview

Figure 12 shows OpenIAM architecture overview where Enterprise Service BUS (ESB) is a central component acting as transit system for carrying data between applications. The heart of system is message bus which routes messages between endpoints. Services provide identity management functionalities such as authentication, authorization, password management, provisioning etc. Services are scalable and extensible for example by ability to plug new methods of authentication.

4.3 OpenIDM

OpenIDM is an open source identity management system written in Java programming language. OpenIDM is flexible, modular and provides RESTful interfaces to satisfy business needs and requirements. System provides password management for defining password policies and also synchronization of passwords from Microsoft Active Directory

(AD) and ForgeRock OpenDJ. OpenIDM offers scalable method for discovering new, changed or deleted accounts.

Architecture is focused on modularity by providing components that can be composed together according to special needs. Component architecture allows to easily add new components or remove existing components. Following figure 13 shows architecture overview on system.

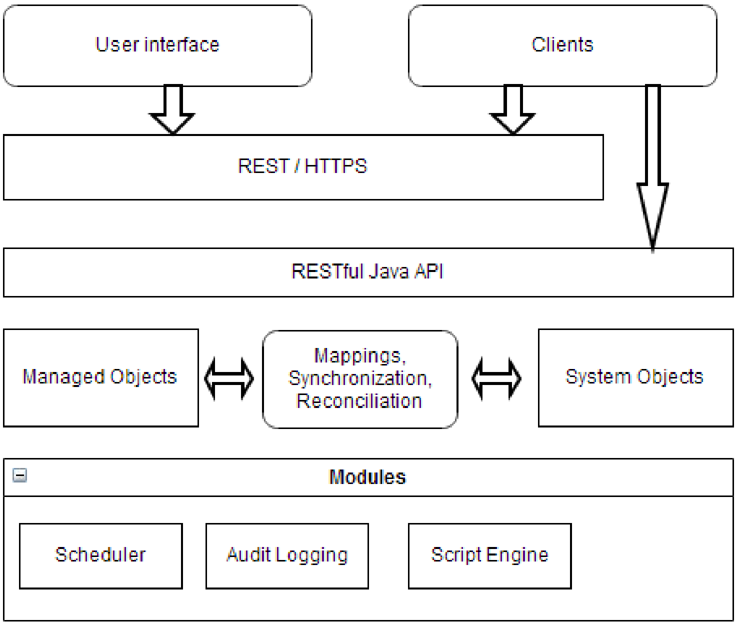


Figure 13. OpenIDM architecture

Core services are for example scheduler that takes care about regular synchronization and reconciliation by using Quartz library. Script engine provides triggers and plugin points for OpenIDM. Audit logging logs all relevant system activity to log stores. It also stores data from reconciliation for reporting. Managed objects represent identity related data managed by OpenIDM. These objects can be configured as user, group or role. System objects are representation of object in external systems. External object for example user entry in external LDAP directory is represented as system object. Mappings define policies between target objects and source objects. Mapping can define triggers for validation, filtering and transformation of source and target objects. Synchronization provides creating, updating and deleting resources from a source to a target system. Reconciliation provides resource comparisons between OpenIDM managed objects and source objects from external systems. Comparisons can result with proper actions depending on defined mapping between systems.

Access layer consist of RESTful interfaces for CRUD operations. User interfaces provide password management, registration and workflow services.

Provisioning system in OpenIDM manages accounts, groups and roles. Provisioning subsystem is connected to other resources systems for example human resource servers, directory servers, provide communication between these systems and take care about managing changes. Changes are propagated by synchronization process that propagates changes from OpenIDM to other external resources or vice versa. There is a chance that inconsistencies arise due to maintenance one of external systems and reconciliation is needed. Reconciliation manages changes by comparing information from external resources and OpenIDM information.

4.4 midPoint

midPoint is an open-source provisioning system providing user provisioning, de-provisioning, synchronization of identities and automated identity management processes. It also supports security and reporting. [26] midPoint solution focuses on efficiency and practical usage. For example provisioning scenarios are easy to setup and use because there is no need to code, instead configuration and simple expressions are needed. midPoint is designed to be modular and extensible by providing open iterfaces and plugins. System core consist of repository component, provisioning and model components (figure 14). Repository is storing authoritative identity data and links to identity objects in other systems, roles and access rights. Provisioning component deal with other systems, read data from them and modifies them if needed. Core components are configurable, but also customizable in special cases (adding new attribute expression or redefinition of a role). Highly customizable component is for example user interface.

midPoint uses hybrid Role-based Access Control model which use rules to extend role definitions so that less roles can handle more situations. midPoint also unifies identity data models from integrated systems providing unified model to reduce integration overhead, but also provides customatizations and exceptions if needed.

Synchronization mechanism of midPoint uses account linking based on correlation and confirmation expressions.

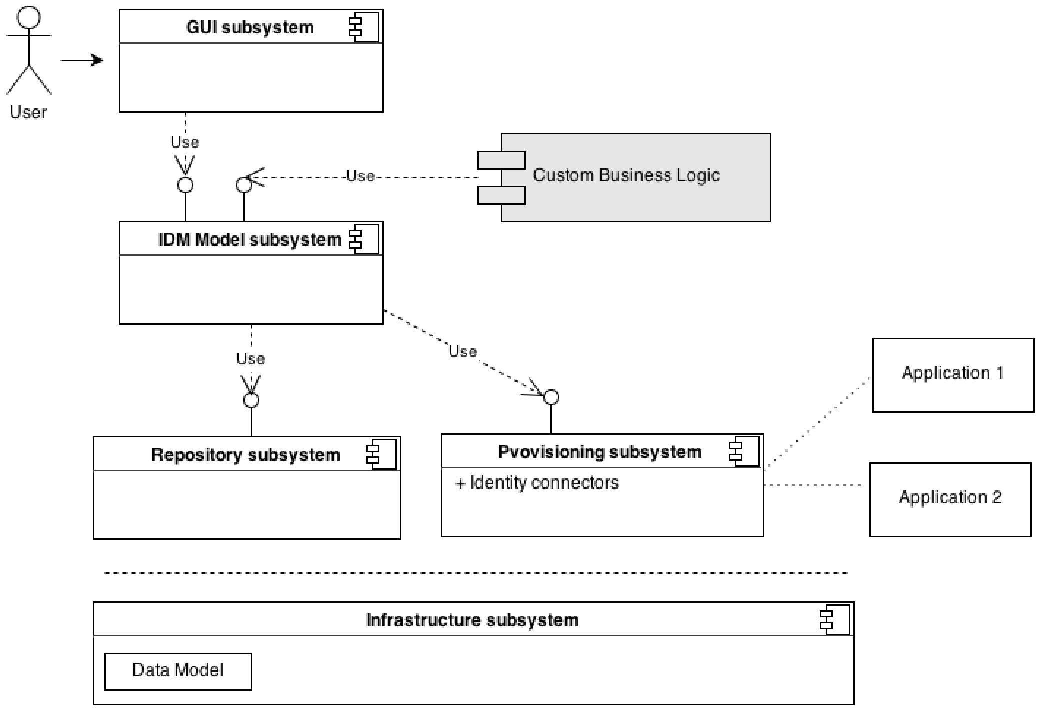


Figure 14. midPoint component overview

5 Correlation method proposal

In this section we would like to present our method proposal for automatic correlation of user records. Our method is based on string similarity metrics detecting similarity of user records and machine learning algorithms for automatic correlation. Method includes these steps (as shown on figure 15)

- Data preparation, normalization of attributes and attribute extraction
- Partitioning records, setting default attribute weights
- Applying appropriate string metrics for similarity estimation
- Applying machine learning algorithms to train model and classification
- Manual checking – verification

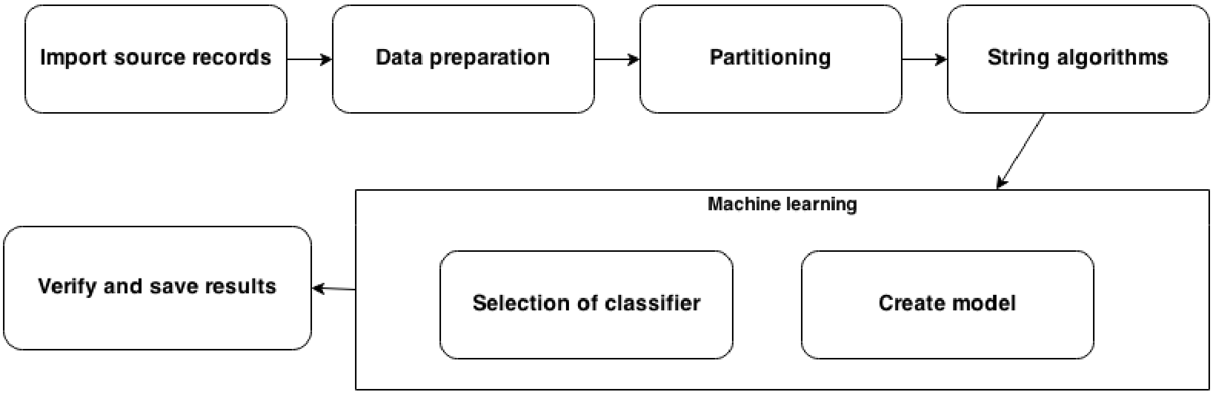


Figure 15. Correlation method overview

5.1 Data preparation and normalization

Data preparation phase is first step to user correlation detection. Input data must be stored uniformly in the database so it is easy to work with them. The problem is that data are stored in heterogeneous structures and so we need to define common structures to work with many data sources. Data preparation deals with parsing input data, transforming and standardize (normalize) them.

Parsing of input data depends on input format for example (xml, csv, database schema, plain text). We designed parser for structure of coma-separated format where records looks like "id: first_attribute_value; second_attribute_value". Transformation of attribute data type is important because we often need to transform for example numbers (integers) to string data type for later processing.

Input data with user account information contain various attributes. For proper matching and correlation detection we must estimate similarities based on attribute values from many sources so there is need to have normalized and standardized attributes. Some of the normalized attributes are suitable for simple string comparison without any other techniques because normalized form is adequate. In most cases, there is need to more sophisticated matching algorithms. The quality of normalized data do not differ from original data so we normalize before insertion into database. There are various methods for data normalization. We apply these basic methods:

- Changing all characters to lower case
- Removing punctuation
- Replacing multiple white space characters with one space character
- Removing white spaces at the beginning of string and in the end of string

For numerical attributes like phone number we remove non numerical characters. For example telephone number “+420 987 343 (2)” is modified to “4209873432”.

5.2 Attribute extraction

Attributes may be too general and represent mixed user attributes for example attribute “name = “Ivan Torna” reflects given name and last name of person. We want to split this attribute and create more specific attributes – given name, last name for more precise string (attribute) matching. For this purpose we propose detection methods which process and detect attributes for Slovak language.

5.2.1 Given name extractor

Given name/first name detector is based on list of given names for specific country or domain (Slovak in our case). It can be modified in order to detect special subset of users (organization has external employees in different country). There is also separate list of male names and female names because of need to detect gender of user. For Slovak users there is another option to detect potential female by finding last name suffix “ová, ova”. The problem with first name is, that when there is misspelling error, we can not find appropriate name from list, so we need to apply string similarity algorithms for approximate estimating of first name.

5.2.2 Last name extractor

Last name detector takes name attribute, remove already detected given name and result is set of potential last names. In most cases, there is only one last name detected, but there is a chance to have woman user with two last names – born last name and marriage last name.

5.2.3 Title extractor

Titles usually take place at the beginning and in the end of full name. As far as we determine given name and last name, we can assume, which strings in name can contain titles. For title detection we use list of honorific titles.

5.2.4 Email address extractor

Email address attribute can be divided into prefix part which usually contains given name, surname or abbreviations of name. Suffix part mostly reflects webmail service (google mail, yahoo mail etc.) or organization domain name (@organization_name.com).

5.3 Partitioning

Finding correlation on large datasets may be too ineffective due to comparing and matching large number of user record objects. In worst scenario, comparing includes Cartesian product. Thus we propose partitioning mechanism, which can make this task more effective (Figure 16). We have user objects with attribute frequencies stored in database sorted by combination of last name attribute frequency and given name attribute frequency. Same approach is applied to source input data, so that we have two sorted list of user objects. Then we choose partitioning strategy of choosing N partitions – experimentally determined

In partitioning we choose sorting attribute with most distinguishing ability. Estimating most distinguishing attribute is based on counting number of attribute values for certain attribute. The more unique attribute values are present within attribute, the more distinguishing ability attribute has. Partitioning algorithm sort two data sources and create data partitions for example data source containing 6000 user records is divided into three partitions of 2000 records. These partitions are used in correlation process where only certain partitions are processed which saves time. Partitioning can be optional when time of correlation process is not critical and then results can be improved.

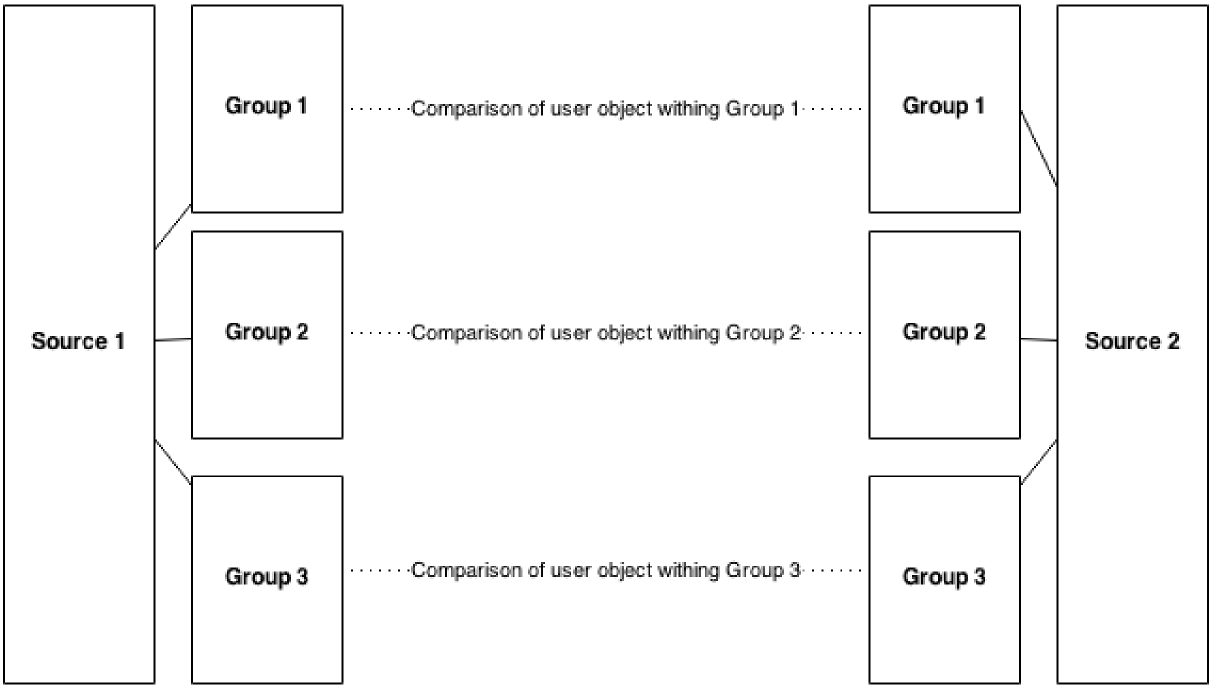

Source 1
Group 1
Group 2
Group 3
Comparison of user object withing Group 1
Comparison of user object withing Group 2
Comparison of user object withing Group 3
Group 1
Group 2
Group 3
Source 2

Figure 16. Partitioning of user records

Our method use string metrics and similarity between records to estimate user correlation. The core of the user correlation problem is that we have one user object, which is compared to other user objects in order to find correlation. We assume set of user objects which are compared to new user objects from other data sources – which are actually duplicates. Then we can use multiplicity of attribute value in set of users to improve similarity metrics and estimations. For example we have 100 users but only one of them has attribute last name set to "Astaloš" and given name to "Jan". This user has unique last name within user set and when estimating new correlation, this attribute have higher distinguishing ability than given name "Jan" which is more often occurring in given set of users. For this purpose we estimate attribute importance for all attributes and choose attribute with highest distinguishing ability to set default weights for string similarity algorithms.

In real life deployment correlations must be verified and so we propose verification tool which summarizes results of correlation and give ability to manually correlate users, or change bad user correlations. Besides overall verification, we want to apply recommendation of potential user matches for manual check in order to improve correlation framework. For example in organization, human resources employee wants to correlate two sources and our system propose him potential correlation during process and he manually approves or rejects our recommendations. We keep this information to improve our correlation model. If there is certain amount of approves without rejects, there will be less correlation proposals. On the other hand, if there are too many rejects, our correlation framework need to apply these facts and change attribute weights, partitioning strategy or string similarity metrics.

5.4 Similarity algorithms

User correlation is based on attribute similarity. We propose basic similarity function *sim$_a$(attr1, attr2),* where each similarity is computed within attribute and between two user objects (UserX, UserY). All similarities are stored in similarity vector [sim$_{a,}$ sim$_{b,}$ sim$_c$ …] and these vectors are used for match estimation. 1 represents match and 0 dissimilarity of compared attributes.

$$sim_a\colon (attr1\ x\ att2) \rightarrow [0,1] \epsilon\ R \qquad (7.1)$$

In general for name attributes we use these distance metrics:

- Jaro distance
- Jaro-Winkler distance
- Hamming distance
- N-gram similarity

We have analyzed multiple string similarity metrics for attributes in our method. Figure 17 shows metrics results for experimental records containing full name attribute, where similarity was estimated between manually modified records (“Marcel Abbas”, “Marcel Abas”). Jaro-Winkler distance, Jaro distance and Soundex were most accurate.

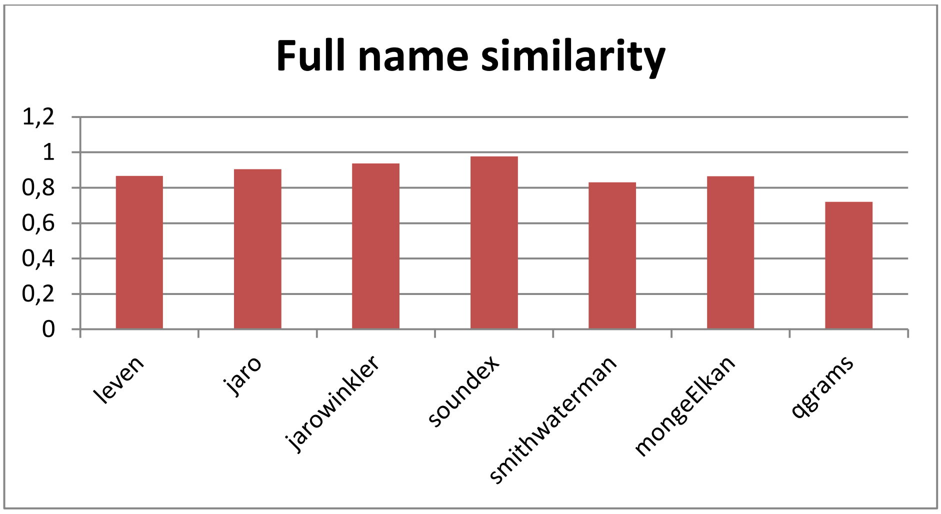


Figure 17. Full name similarity results

We have also analyzed all other attributes (given name, last name, email, organization unit etc.) from our dataset for most efficient and accurate string similarity metric.

For given name attribute we use Jaro-Winkler distance, where first characters of string are more important which makes given name comparison more effective because of low number of misspellings at the beginning of given name. More misspellings occur in the middle of strings.

For last name attribute we use Jaro distance, which is suitable for short strings. Jaro distance count with length of string so that misspelling in longer word is less important than in shorter word. Jaro distance and Jaro-Winkler distance are quite similar, so we want to experiment combinations of attributes and different distance metrics. Misspellings in names occur in the middle of string and so we can use Hamming distance in case of same length of compared strings. Distance counts number of different characters which is suitable for strings of same length. N-gram similarity split string into n-grams for example N=3 (“tomas” = “tom”, “oma”, “mas”). Jaccard coefficient is applied to results of n-gram similarity. There is problem with estimating suitable value of N, so that the substrings are not too short or too long.

Honorific titles and titles are compared by using Hamming distance and Damerau-Levenshtein distance which is similar to Levenshtein distance but there is extra operation – transposition (swapping characters). This distance is good for short strings and strings with misspellings, which can easily occur in titles and honorific titles.

E-mail attributes are compared by Hamming distance for e-mail suffixes and Jaro and Jaro-Winkler distance for email e-mail prefixes.

Organization unit attributes are usually longer strings or abbreviations. Therefore we need to combine Levenshtein distance, N-gram distance and Damerau-Levenshtein distance.

Numeric similarity metrics are usually based on simple string conversion and primitive comparison. We propose cosine similarity for numerical attributes like age etc.

5.5 Machine learning algorithms

Our method prepares source data for supervised machine learning phase, which can automatically classify records. The training data consists of training examples vectors containing similarities between each record attributes and label (if records are duplicates or not). Machine learning algorithms analyzes these training data and creates inferred function – model for mapping new examples (Figure 18). The main advantage of supervised machine learning is that once the model is trained, we can apply it to various datasets. Our method:

1. Partition source data
2. Creates similarity vectors between records
3. Creates training set
4. Apply learned model to example set

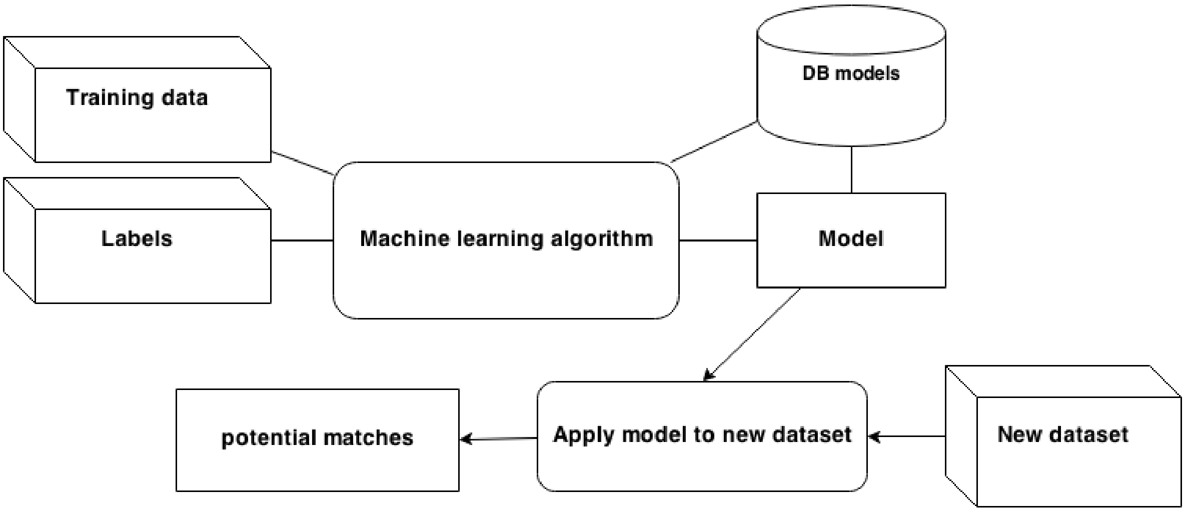


Figure 18. Machine learning process overview

We analyzed and applied these machine learning algorithms:

1. Support vector machine (SVM)
2. K-nearest neighbour (KNN)
3. Logistic regression (LR)
4. Neural network (NN)
5. Decision trees

For each algorithm there are parameters to be set which were tuned in our experiments.

6 Correlation framework

We propose complex framework for correlation process of user identities from many heterogeneous sources. Our goal is to make process as automatic as possible in order to save time with manual correlation. We deal with data preparation, normalization and we are proposing detection methods for attribute enrichment. Our correlation method is based on similarity metrics and machine learning algorithms. The scenario of usage correlation framework (Figure 19):

- User choose data source
- User imports data source in CSV format (comma separated values) and choose either automatic attribute weights or manually set attribute weights
- User records are automatically sorted and split into partitions
- User starts automatic correlation process
- User see results of correlation process and verify matches
- Model is adjusted if results are incorrect

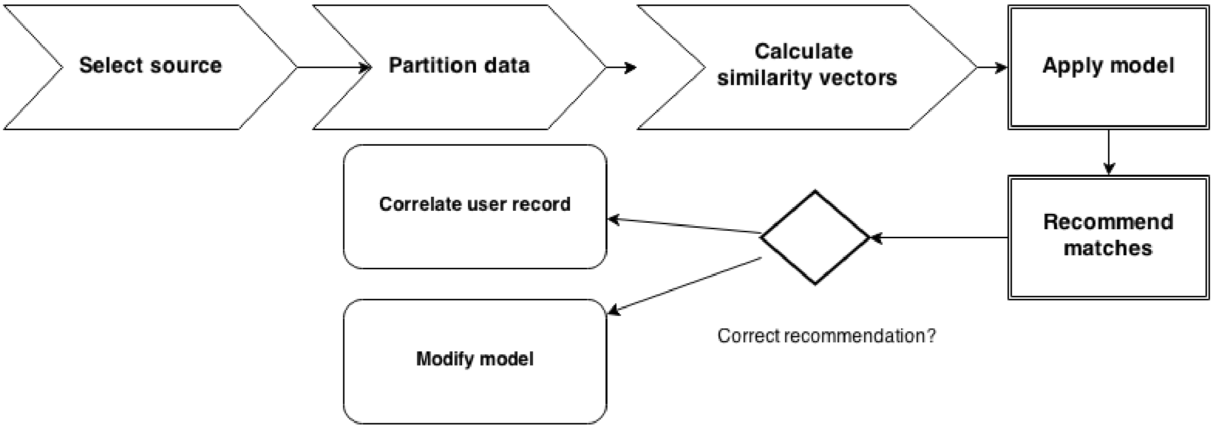


Figure 19. Correlation framework schema

Input data are pre-processed and normalized. Then creating user objects takes place. For every record in database there is user metadata object with default weights and attribute frequencies stored in user metadata object. User metadata object may also contain multiple attribute values for certain attributes from preceding source correlation detection. User objects are partitioned and for each user in partition, matching vectors are created:

$$\textit{Attribute vector} = [attr_1, attr_2 ... attr_n]$$

$$\textit{Default weights vector (DW)} = [dw_1, dw_2 ... dw_n]$$

Similarity between same attribute *i* of two user objects is defined as:

$$sim(a,b)_i = D_i . \frac{DW_{ai}+DW_{bi}}{2} \tag{6.2}$$

D_i represent distance between attribute$_i$ values computed by using similarity algorithms between two user objects. DW_i is experimentally predefined value (Formula 6.2). Overall similarity between two user objects *a,b* is:

$$sim(a,b) = \frac{\sum_1^N sim(a,b)_i}{N} \tag{6.3}$$

Where *N* represents number of attributes in user object (Formula 6.3). Result of similarity estimation is classified by using K-nearest neighbor algorithm, logistic regression, decision tree and Support Vector Machine (SVM) classifiers.

Result data are split into training set and testing set. Training set contains manually pre-labeled tuples which are used to train classifier. Testing set is used to validate learned classifier.

Trained classifiers can be applied on real datasets and user sources. Adding single source of data and integration with existing database need to implement updating mechanism for database records and user metadata objects. When new record is inserted, frequencies must be re-calculated. If user match was detected, we need to store attribute values in metadata object, so that some of the attributes may differ, or can be enriched by new record. For example when user A has e-mail “jan@gmail.com” and is correlated to user A’ which has e-mail “jan.sukenik@gmail.com”, we want to keep this information for future matching. Here is overall schema of our framework (also shown on figure 20):

- Create source object and import data (comma separated format)
- Pre-processing transforms and normalize input data by using detection methods for attributes
- User objects are created (When database is empty, attribute frequencies must be computed)
- Order data by attribute (most distinguishing)
- Partitioning data (Only same partitions are compared with their estimated similarities)
- Similarities between records are computed
- Creation of training and testing set
- Classifier is trained
- Model is applied to example set
- User verifies results

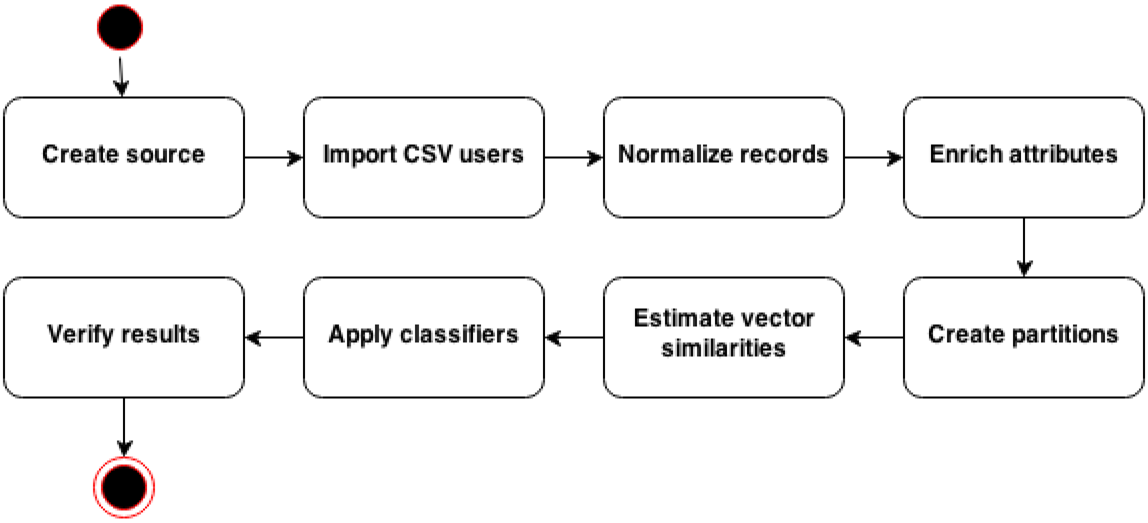


Figure 20. Correlation framework prototype schema

We propose enriched user schema model with ability to store multiple attribute values for one user. If we detect correlating user objects, some of their attribute values can vary and so we want to keep this information for future matching. For example we have two correlated user objects – U_1, U_2 where $U_{1(email)}$=”martin.svec@gmail.com”, $U_{2(email)}$=”m.svec@gmail.com”. We keep both email addresses in database for improvement matching other future data sources.

6.1 Dataset

Suitable dataset with appropriate organizational and personal data with Slovak people was not easy to find, but we found dataset containing employees of Slovak University of Technology. This dataset contains 6625 users with personal and organizational attributes:

- Full name
- Organizational unit
- Office number
- Telephone
- E-mail

User record example: “*Ing. Marta Ambrová, PhD.;OAT ÚATM FCHPT;SB 172;+421 (2) 59 325 783;marta.ambrova [at] stuba.sk*”

We created smaller subsets of original dataset which contains (50, 100, 150, 200, 500) randomly selected users. These subsets are manually modified and misspellings to attribute values are created, abbreviations and character swaps are applied. The reason why we need to create subsets manually is fact, that there are no sufficient sources of public available data on users or employees with Slovak people. String similarity algorithms were tuned on subsets of these data for example tuning of given name for Jaro-Winkler algorithm was applied on list of given names from STU dataset.

6.2 Implementation

Our correlation framework is implemented in ruby language and web framework Ruby on Rails. We use

- PostgreSQL database system
- Libsvm library (Library for support vector machines)
- Ai4r (Collection of ruby algorithms for classification and clustering)
- Liblinear (Logistic regression)
- RubyFann (Library for neural network classifier)
- Decision tree library for ruby
- Amatch, Fuzzy string match, Hotwater (Libraries for string similarity algorithms)

Correlation framework is designed as web application. We have chosen Ruby language due to its easy and clear syntax, strong community support, availability of various libraries. Ruby on rails is web based framework which makes web development as easy as possible so we can focus on functionality. Our correlation framework was designed as web application because nowadays trend is to provide software as a service. User trough web interface access functionality so user do not need to install and maintain system locally. Input of our system is CSV format because it is common and simple form. It is also easy to process input data. Moreover another formats like XML or JSON can be easily transformed to CSV format. We store all data in PostgreSQL relational database because it is free, reliable, scalable and stable with strong support. Relational database was chosen because of need for ad-hoc queries based on filtering various columns, easy to use SQL syntax and it’s maturity (stability, bug free, well tested over years).

7 Experiments

For evaluating results of classifiers, precision, recall and F_1 measure are used. Precision is the ratio of number of relevant records retrieved to the total number of irrelevant plus relevant records (Formula 7.1). Recall (Formula 7.2) is the ratio of the number of relevant records to the total number of relevant records. Both precision and recall are expressed as percentage. Metric that combines the precision and recall of the metric in the harmonic mean is called F-measure (Formula 7.3), also known as the F1 metric. F1 take values from interval <0,1> and the higher value is, more successful system is. Since the F-metric is a combination of precision and recall, it corresponds to a compromise between accuracy and coverage. Results can be divided into four classes:

- **True positives** correct matches
- **True negatives** correct non-matches
- **False positives** incorrect matches
- **False negatives** incorrect non-matches

$$P\ (precision) = \frac{true\ positives}{true\ positives + false\ positives} \tag{7.1}$$

$$R\ (recall) = \frac{true\ positives}{true\ positives + false\ negatives} \tag{7.2}$$

$$F1\ \ = 2\frac{P.R}{P+R} \tag{7.3}$$

7.1 String similarity metrics and attribute weights

Correlation based on string similarity metrics without machine learning is first step, where we tuned our detection methods and similarity algorithms for attributes. We have also set up default weight estimation and partitioning strategy.

Default weights are dependent on distinguishing ability of attribute so that default weight for more distinguishing attribute is higher. We used example set of 150, 200, 6000 randomly chosen user records from our dataset to estimate which attributes should have higher default weights for string similarity estimation and results are averaged.

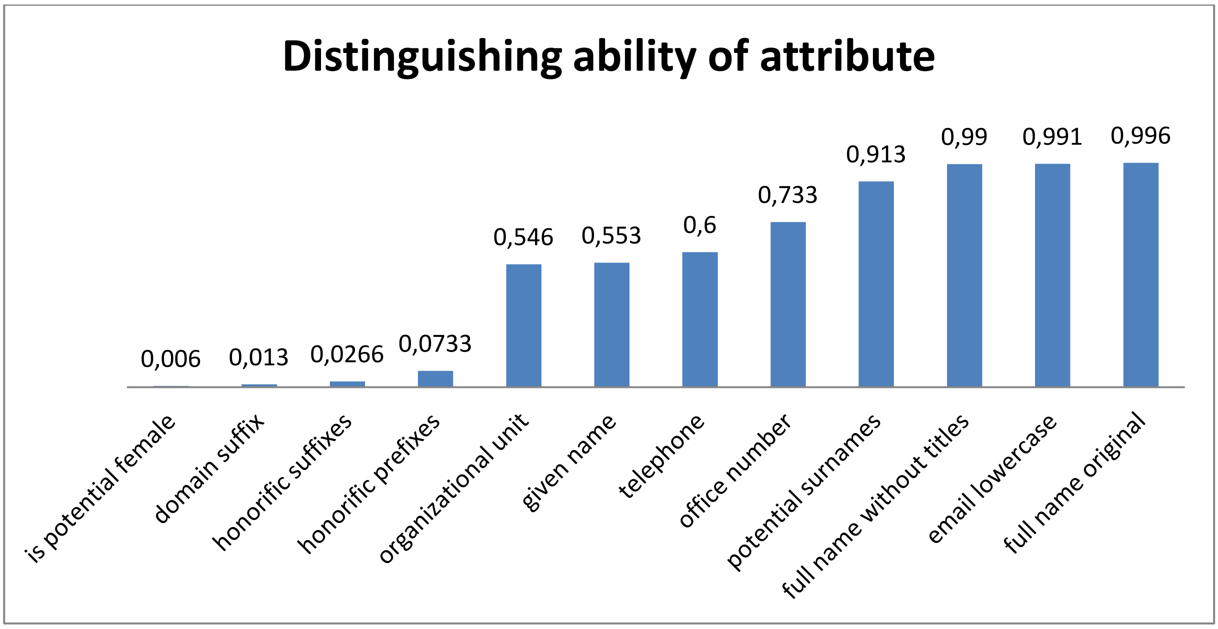


Figure 21. Attribute distinguishing ability overview

As shown on figure 21 the most distinguishing ability has attributes:

- Full name original
- Email (in lowercase)
- Full name without titles
- Potential surnames
- Office number

Default weights are computed as sum of unique attribute values divided by number of records.

Default weights for all attributes are used as default weight vector in similarity estimation between two user records.

To verify the impact of using weights of attributes in similarity matching, we have made experiment with comparisons of small sample data with and without weights. We have randomly chosen 150 user records from dataset and then we have modified their attributes randomly. For every record there is one or more manually created errors (misspellings, omitting parts of honorific titles, abbreviations etc.). The process consisted of these steps:

- Import sample user data
- Estimate default attribute weights
- Compare records and create similarity vectors
- Adjust boundary for classification
- Manually check results of classification and repeat process without default attribute weights

As shown on figure 22, overall results are slightly better with using default attribute weights. Precision is improved by 8% and F1 measure is improved by 4%. String similarity method with manually adjusted boundary for classification is good for experimenting and improving correlation method, but can not be used in automatic classification.

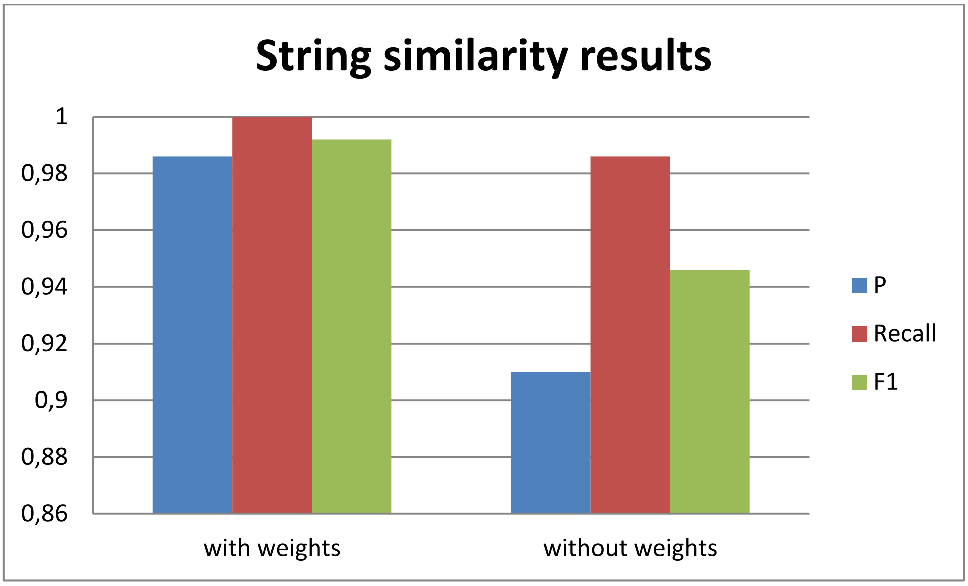


Figure 22. String similarity with and without attribute weights

String similarity algorithms were individually tested on STU users dataset and distance metrics with higher score for each attribute are:

- Full name original – Jaro-Winkler distance
- Given name – Jaro-Winkler distance
- Full name without titles – Jaro-Winkler distance
- Honorific prefixes – Levenshtein distance
- Honorific suffixes – Levenshtein distance
- Potential surnames – Jaro
- Generat TEXT attribute – Ngram (N = 2)

- Number – Jaro
- Domain suffix – Jaro-Winkler distance
- Email lowercase – Levenshtein distance
- Email prefix – Jaro-Winkler distance

7.2 Machine learning algorithms comparison

We have experimented with multiple machine learning algorithms and various parameter settings. First we applied KNN (k nearest neighbour) algorithm and SVM (Support vector machine) with using default weights and without data partitioning. We have experimented on dataset consisting of 1000 user records in which 150 records contained one or more misspellings or errors. For KNN algorithm we experimented with parameters:

- Number of neighbours (K=5, K=10, K=30)
- Distance metric (Euclidean distance, Cosine similarity)

Parameters for SVM algorithms are:

- Type of kernel function (Radial, DOT, Polynomial)
- Epsilon
- C (Soft margin)

SVM achieved better results. SVM has F1 score about 3% higher than KNN. Number of user vecetor comparisons is 1000 000, so that SVM classification took approximately 2 times longer than KNN classification (7 minutes). Despite simple KNN implementation and need for tune only two parameters, we choose SVM classifier for future experiments because results quality matters more than time in our case.

Our second experiment compared SVM, Neural Networks and Logistic regression. We used dataset containing 150 user original user records and 150 misspelled records. We have tested results with partitioning and without partitioning. For SVM classifier we used tuned parameters from first experiment and for Logistic regression the parameters are:

- C
- Epsilon
- Solver type
- Bias

For Neural Network classifier we were not able to tune classifier parameters and results were inadequate (0.4 F1 score). The problem was proper setup of neurons in hidden layer. Input data – vectors are Cartesian product and so that most of the vectors are not matches so that network function was not able recognize patterns. Results for SVM and logistic regression are above 0.9 F1 score (Figure 23). Results show that logistic regression has higher F1 score than SVM. The score is even higher with partitioning strategy (In our experiment 150 user records were partitioned in group of 50 records – 3 partitions) as shown on figure 24. Time of classification was approximately same with SVM and logistic regression.

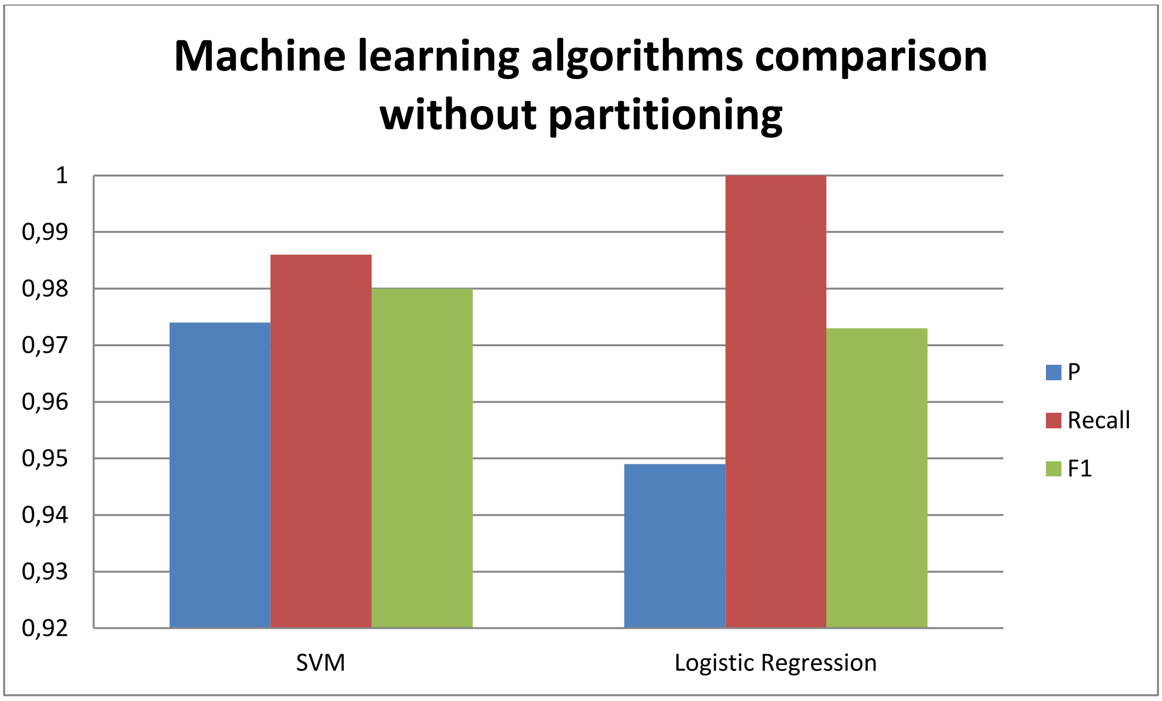


Figure 23. SVM vs. Logistic regression comparison

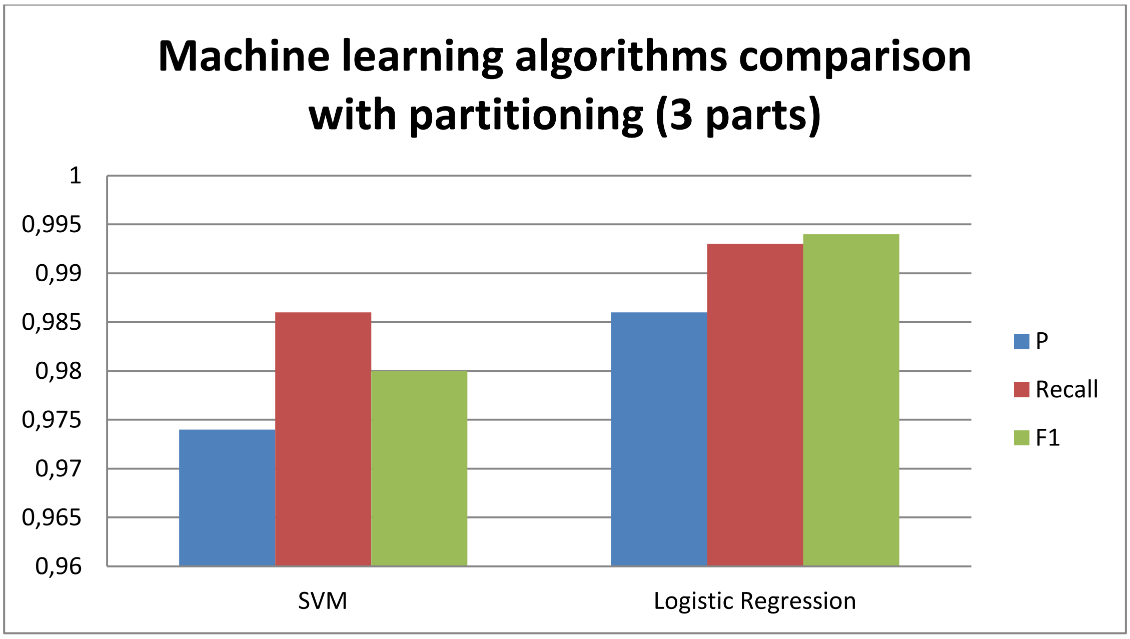


Figure 24. SVM vs. Logistic regression comparison with partitioning

Learning curve for linear regression (Figure 25) and SVM classifier (Figure 26) trained on dataset consisting of 500 user records in which 30 records were modified and order of records was shuffled shows that the bigger training set is, lower error rate is. Error rate in this experiment is ratio of misclassified (FP, FN) records to all records.

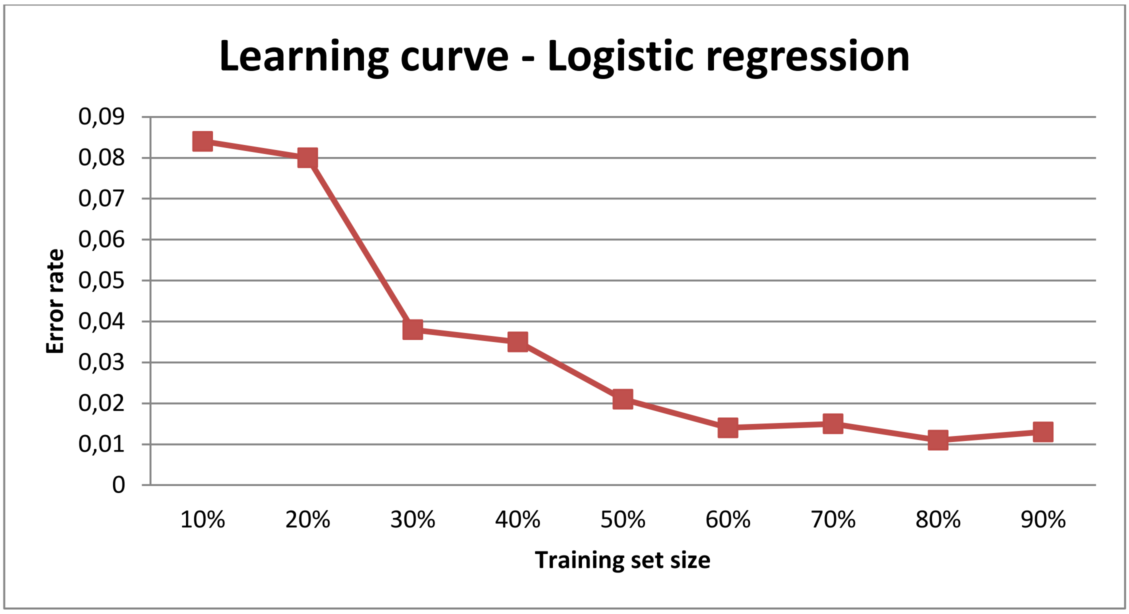


Figure 25. Learning curve for logistic regression

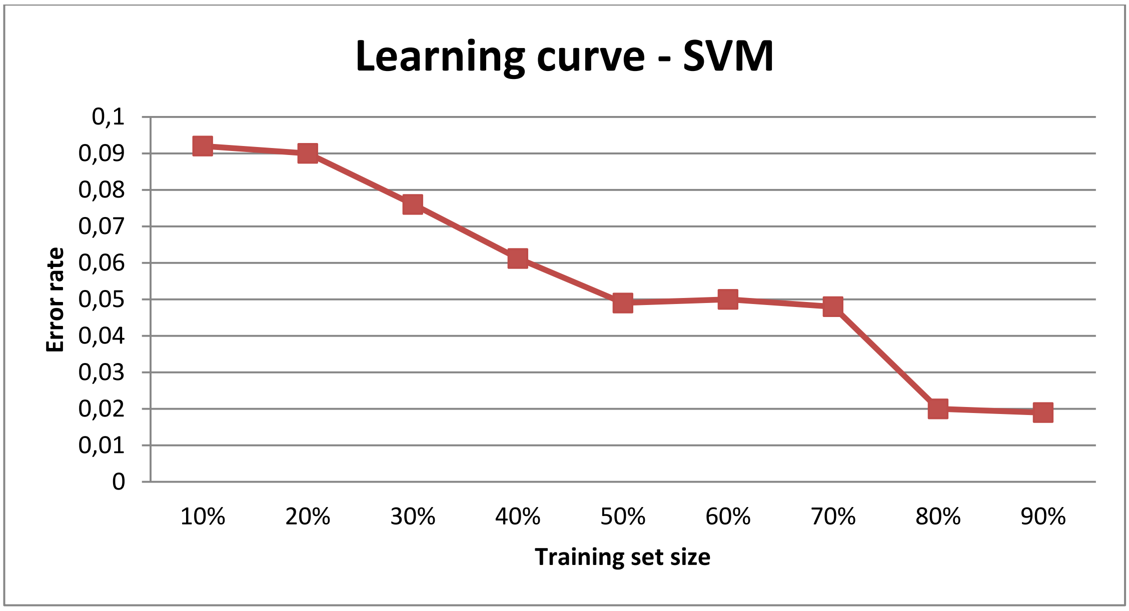


Figure 26. SVM vs. Learning curve for SVM classifier

The partitioning of input data is important for classification larger datasets, because learning and applying model for 6000 records (36 000 000 vectors) may take ~ 2,5 hour. We have tested partitioning size for SVM classifier where we split user records to groups of 10%, 20%, 30%, 40%, 50% and above 50% there are only two groups. With 150 user records, 10% split is 15 user records. Initial 150*150 (22500 comparisons) is reduced to the 2250 comparisons. The smaller the group is, the quicker the classification is, but error rate is higher. In small group there is high chance of missing records. On figure 27 there is error rate for SVM classifier with partitioning that shows low error rate at 40-50%. It means that when we have 2 or 3 groups the error rate is low.

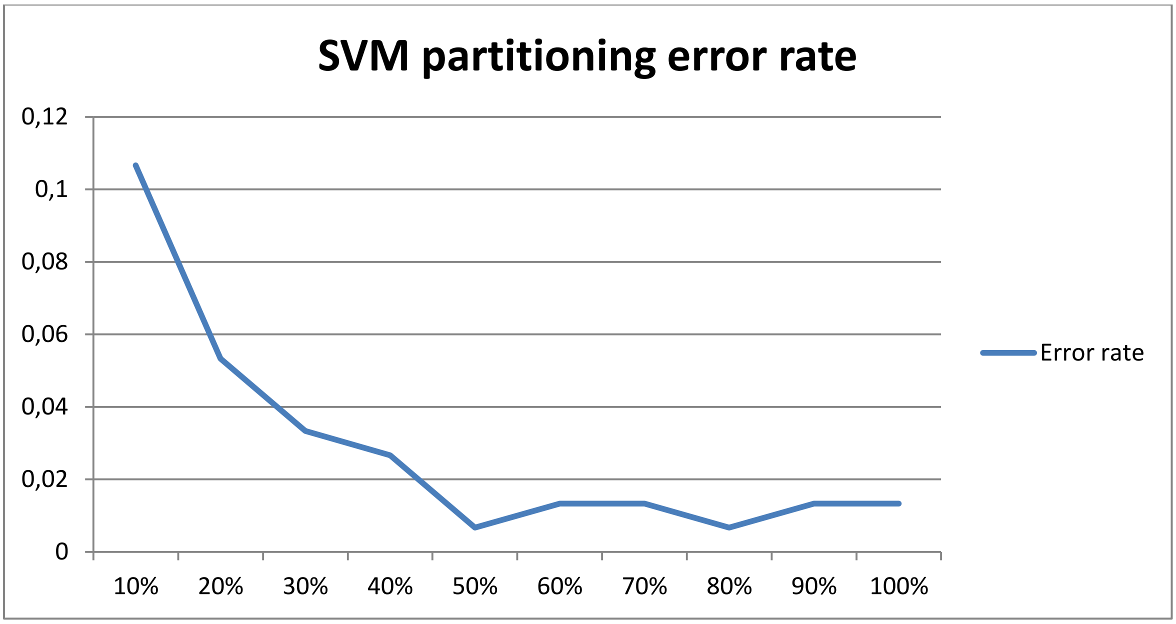


Figure 27. SVM classifier error rate with partitioning

Another factor in correlation detection is time of creating model (training and applying) in order to partition size. On figure 28 there are four classifiers and three partition sizes. Classification process takes up to 100 seconds for dataset (150 user records) for SVM, Logistic Regression and Neural network. For decision tree the time grows much more faster than for others. 60% partition size equals 11700 vectors in training set and Decision tree classification took 900 seconds.

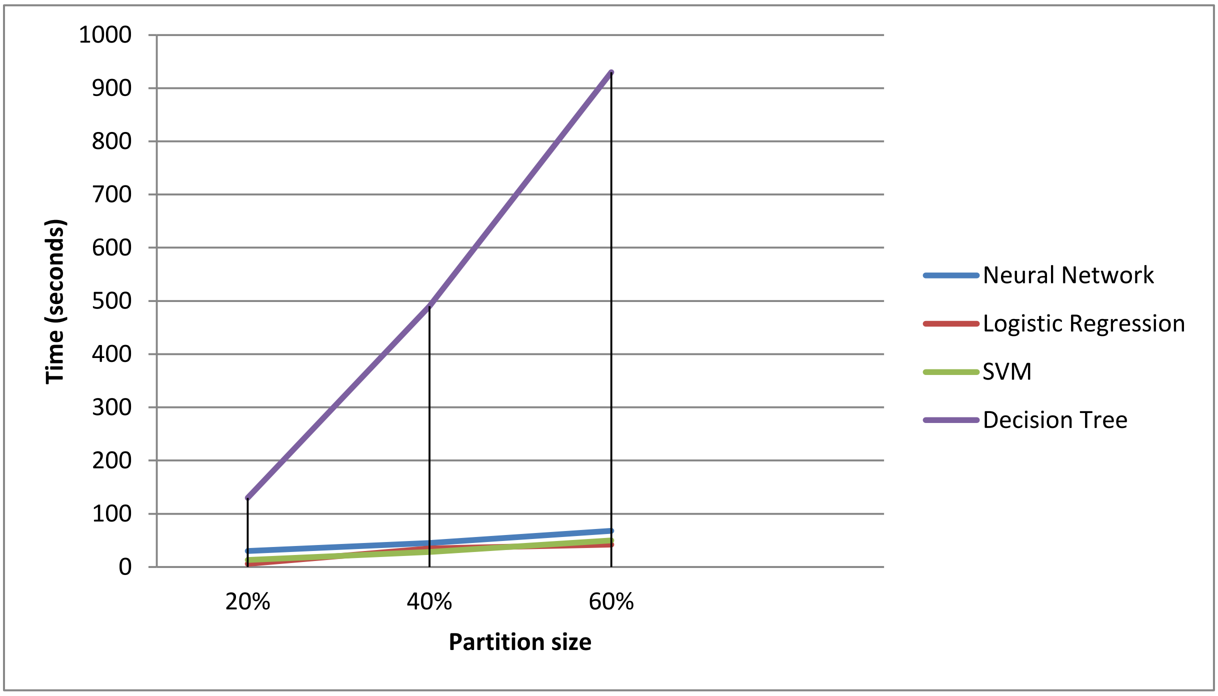


Figure 28. Time of classification and partitioning size dependency

In our last experiment (Figure 29) we created dataset consisting of 150 user records and classify them using:

- Logistic regression
- SVM
- Decision tree

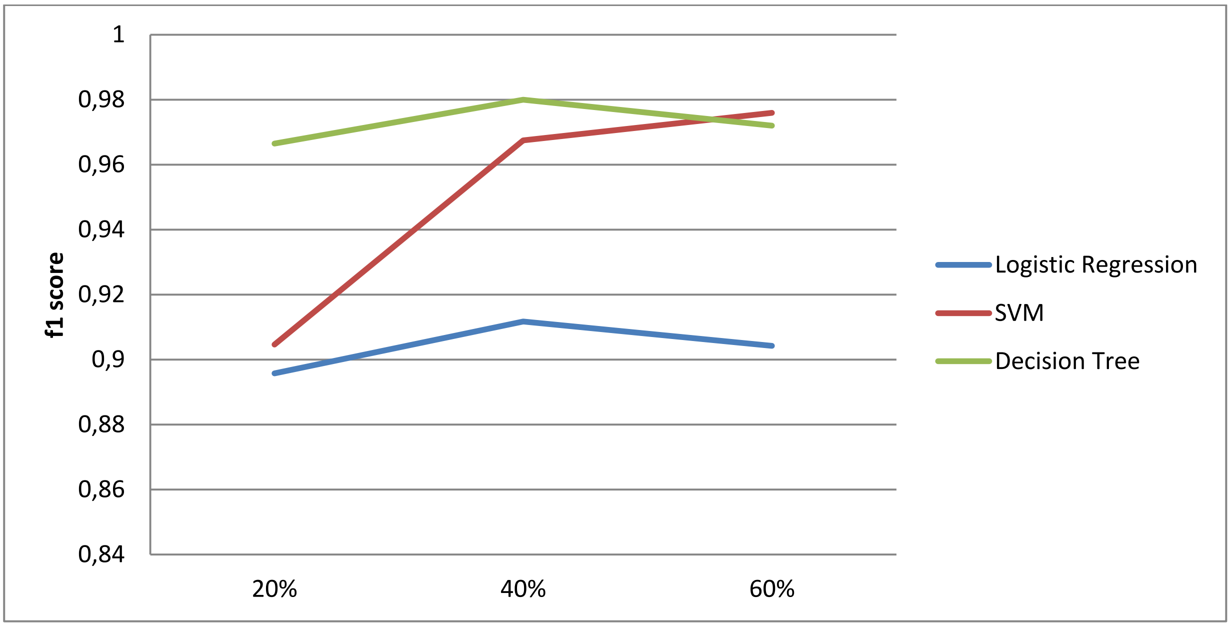


Figure 29. Classification score

Decision tree has higher F1 score in all partition cases, but from the previous experiment the time of classification is much higher than others. In large datasets it could cause problems. SVM classifier has better score when partitioning size is between 40 – 60 %.

7.3. Experiment summary

We did experiments with string similarity metrics where metrics with best score were selected and implemented in framework as part of creation of similarity vector between two records. We have implemented machine learning algorithms – SVM, neural network, logistic regression and decision tree. SVM and logistic regression gained highest score and are suitable for experimental dataset.

8 Conclusion

In our work we analyzed field of identity management systems with focus on provisioning and identity stores which manages identity lifecycle and identity storage within organization trough identity management system.We have analyzed possibilities of correlation detection as a part of record linkage field within identity management systems. Particular focuse was on automatic identity correlation process, which includes cleaning and standardization of input data sources, finding similarities between records and classification in order to make correlation proces as automatic as possible.

We have discovered that existing open identity management systems deal with correlation of users mainly manually, but there are solutions like correlation expressions, where some rules are applied to make this process more automatic thus these are not sufficient. String similarity metrics are widely used in deduplication problems, but they need to be properly setup when dealing with identity correlation detection. Even more important in automatic correlation process are machine learning algorithms which discover regularities and classifies data.

The main goal of our work is to design effective and automatic correlation method for identity correlation detection. The method is part of our correlation framework implemented as web application.

Our method deals with data preparation, normalization of input identiy data. We designed attribute extraction methods which process normalized input data in order to improve similarity detection between two records and creates specific sub-attributes for example full name attribute is divided into given name, surname, honorific prefixes etc. When working with large amount of identity data, we need to create partitions which are mutually compared in order to improve time difficulty. Our correlation method is based on similarity estimation between two records, where each attribute is compared and similarity vector is computed. We propose mechanism for estimating default attribute weights based on attribute distinguishing ability so that similarity vector can be computed with these weights. User can adjust these weights if needed. Our method uses experimentally selected string metrics for each attribute type. Final correlation is proceed by using machine learning algorithms such as SVM and Logistic regression and final results are verified manually by user. User can use predefined classification models, or can create own model.

The method was verified by using our correlation framework, where experimental dataset was imported, processed, classifiers were trained and applied to testing set. Results were manually verified. We have made various subset from original dataset and adjusted parameters for classifiers in order to impruve score.

We have realized experiments with string algorithms, partitioning strategies and machine learning algorithms.

Our method is part of correlation framework which can process various input data. Supported format is CSV „comma separated values“, but can be easily extended to other formats i.e. XML, JSON etc. User can create own classification model and apply to testing set, but there is need to create training set himself. User can adjust classification parameters in order to improve score and use this model to new data sources.

References

[1] Andreas Pfitzmann, Marit Hansen. Anonymity, Unlinkability, Unobservability, Pseudonymity, and Identity Management – A Consolidated Proposal for Terminology. 2006

[2] Sheila Tejada, Craig A. Knoblock, Steven Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining,* pages 350-359, 2002

[3] Mikhail Bilenko, Raymond J. Mooney. Adaptive Duplicate Detection Using Learnable StringSimilarity Measures. In *proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining,* pages 39-48, 2003

[4] Sebastion Clau ß, Dogan Kesdogan, Tobias Kolsch. Privacy Enhancing Identity Management: Protection Against Re-identification and Profiling. In *proceedings of the 2005 workshop on Digital identity management*, pages 84 – 93, 2005

[5] Audun Jøsang, Muhammed Al Zomai, Suriadi Suriadi. Usability and privacy in identity management architectures. In p*roceedings of the fifth Australasian symposium on ACSW frontiers - Volume 68,* pages 143-152, 2007

[6] Semančík, R. Choosing the Best Identity Management Technology for Your Business. In *proceedings of InfoSecOn 2006 Conference, Cavtat, Croatia*, pages 1- 10, 2006.

[7] Tarik Mustafic, Arik Messerman, Seyit Ahmet Camtepe, Aubrey-Derrick Schmidt, ´ Sahin Albayrak. Behavioral biometrics for persistent single sign-on. *In proceedings of the 7th ACM workshop on Digital identity management*, pages 73 – 82, 2011

[8] Messaoud Benantar. Access Control Systems Security, Identity Management and Trust Models. Available from http://shazkhan.files.wordpress.com/2010/10/access-control-systems.pdf

[9] Li Hui, Shen Ting. A token-based single sign-on protocol. In *proceedings of the 2005 international conference on Computational Intelligence and Security - Volume Part II,* pages 180-185

[10] Jani Hursti. Single Sign-On. Available from http://www.tml.tkk.fi/Opinnot/Tik-110.501/1997/single_sign-on.html

[11] D. Ferraiolo, J. Cugini, R. Kuhn."Role Based Access Control: Features and Motivations. *In p*roceedings, Annual Computer Security Applications Conference, *IEEE Computer Society Press*, 1995

[12] David F. Ferraiolo and D. Richard Kuhn. Role-Based Access Controls. *Reprinted from*
15th National Computer Security Conference (1992) Baltimore, pages 554 – 563, 1992

[13] Joseph Pato. Identity Management: Setting Context. Available from http://artemis.cs.yale.edu/classes/cs155/spr03/idmgmt-tr.pdf

[14] Axel Buecker, Dr. Werner Filip, Jaime Cordoba Palacios, Andy Parker. Identity Management Design Guide with IBM Tivoli Identity Manager. Available from http://www.redbooks.ibm.com/redbooks/pdfs/sg246996.pdf

[15] William E. Winkler. Record Linkage Software and Methods for Merging Administrative Lists. *STATISTICAL RESEARCH REPORT SERIES NO. RR/2001/03, WASHINGTON DC, US BUREAU OF THE CENSUS,* 2001

[16] Indrajit Bhattacharya, Lise Getoor. Iterative record linkage for cleaning and integration. In *proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery,* pages 11-18, 2004

[17] Alvaro Monge , Charles Elkan. An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records.

[18] Peter Christen. A Comparison of Personal Name Matching: Techniques and Practical Issues. *In proceedings of the Sixth IEEE International Conference on Data Mining – Workshops,* pages 290-294, 2006

[19] Jun Gong, Lidan Wang, Douglas W.Oard. Matching Person Names through Name Transformation. In *proceedings of the 18th ACM conference on Information and knowledge management,* pages 1875-1878, 2009

[20] Peter Christen. Development and user experiences of an open source data cleaning, deduplication and record linkage system. In *ACM SIGKDD Explorations Newsletter Volume 11, issue 1,* June 2009, pages 39-48

[21] Qun Ni, Jorge Lobo, Seraphin Calo, Pankaj Rohatgi, Elisa Bertino. Automating Role-based Provisioning by Learning from Examples.In p*roceedings of the 14th ACM symposium on Access control models and Technologies,* pages 75-84, 2009

[22] Sun Identity Manager Deployment Guide. Available from http://docs.oracle.com/cd/E19225-01/820-5820/ahucl/index.html

[23] Leicester Gill, Great Britain. Office for National Statistics. Methods for automatic record matching and linkage and their use in national statistics, 2001

[24] Mark A. Hall. Correlation-based Feature Selection for Machine Learning, 1999

[25] David W. Aha. Case Based Learning Algorithms

[26] midPoint community. Project wiki page: Architecture.
https://wiki.evolveum.com/display/midPoint/Architecture+and+Design

[27] K. Zeilenga, "Lightweight Directory Access Protocol (LDAP)", RFC 4510, June 2006. [Online]. Available: http://tools.ietf.org/search/rfc4510

[28] J. Kohl, C. Neuman, "The Kerberos Network Authentication Service V5", RFC 1510, September 1993. [Online]. Available: http://www.ietf.org/rfc/rfc1510.txt

[29] Dan Jurafsky. Natural Language Processing, 2012. [Online]. Available: http://www.nlp-class.org

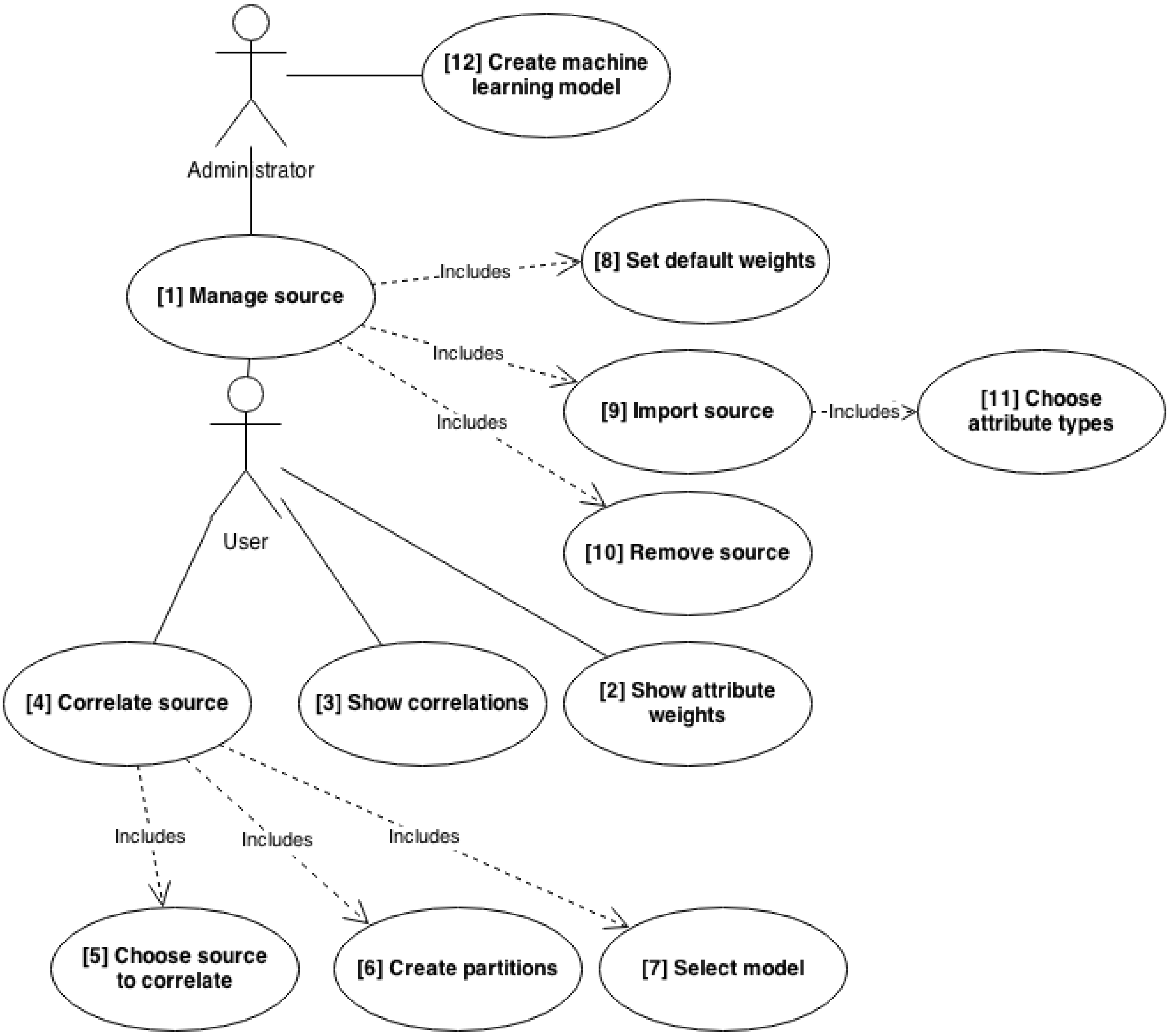
[30] Wikipedia. Damerau-Levenshrein distance, 2012. [Online]. Available:
http://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance

[31] Wikipedia. Knuth Morris Pratt algorithm, 2013. [Online]. Available:
http://en.wikipedia.org/wiki/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm

[32] M Narashima Murty, V. Susheela Devi, 2011, Pattern Recognition.

Appendix content

A Use case diagram



UC1 Manage source

- Interface for managing source

UC2 Show attribute weights

- List of attributes
- Each attribute has own attribute weight
- User can change attribute weight

UC3 Show correlations

- List all matches from correlation process for source
- Remove match from database

UC4 Correlate source

- Interface for correlation process

UC5 Choose source to correlate

- User selects source to correlate with actual source

UC6 Create partitions

- Source can be partitioned
- User inputs group size

UC7 Select model

- User can see list of predefined models
- User can choose appropriate model from database
- Apply selected model

UC8 Set default weights

- Default weights are computed for all records

UC9 Import source

- User select CSV file to import
- User choose attribute types for each column

UC10 Remove source

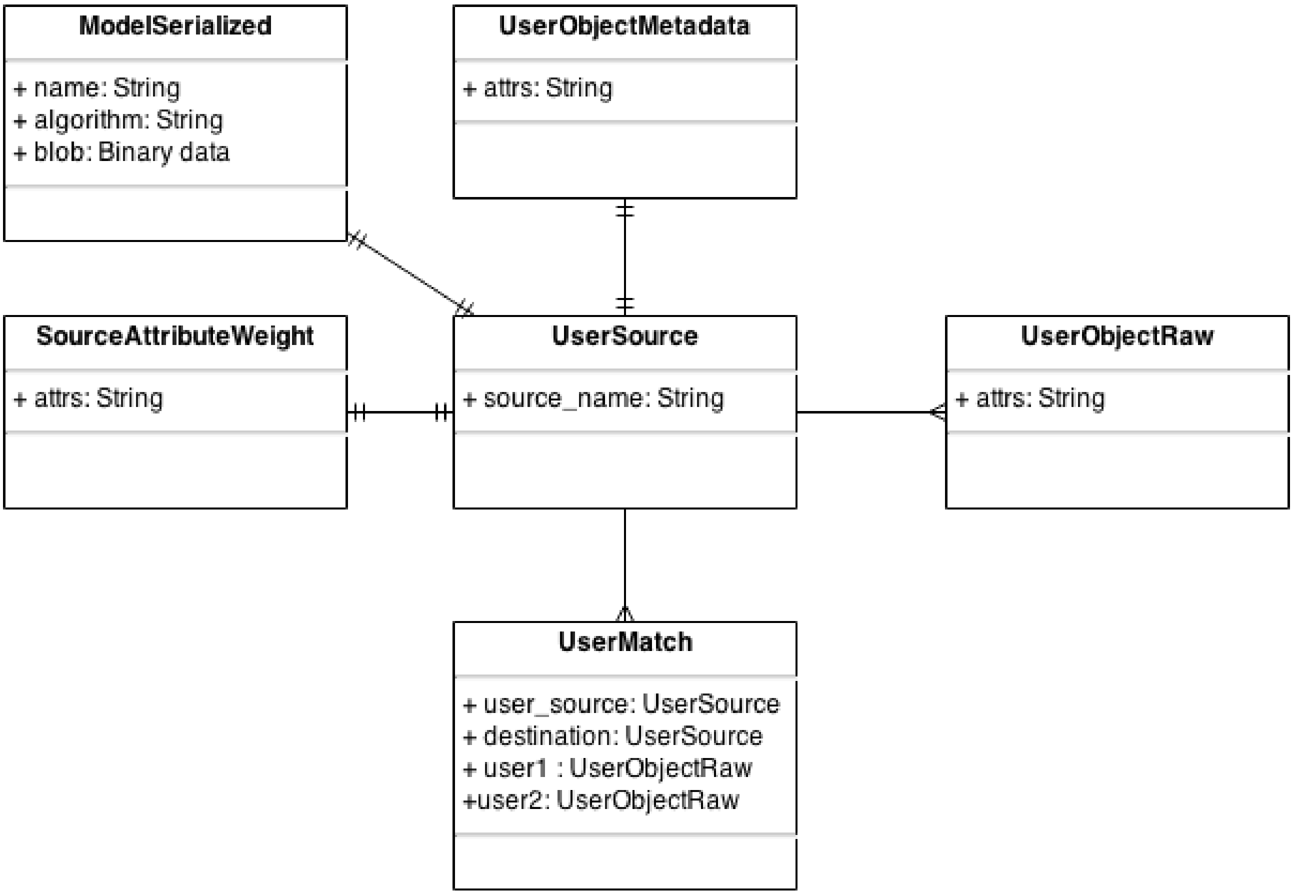
- Source is removed from system
- All records and metadata objects are removed

UC12 Create machine learning model

- Select machine learning algorithm
- Choose partitioning size for group
- Set parameters for machine learning algorithm

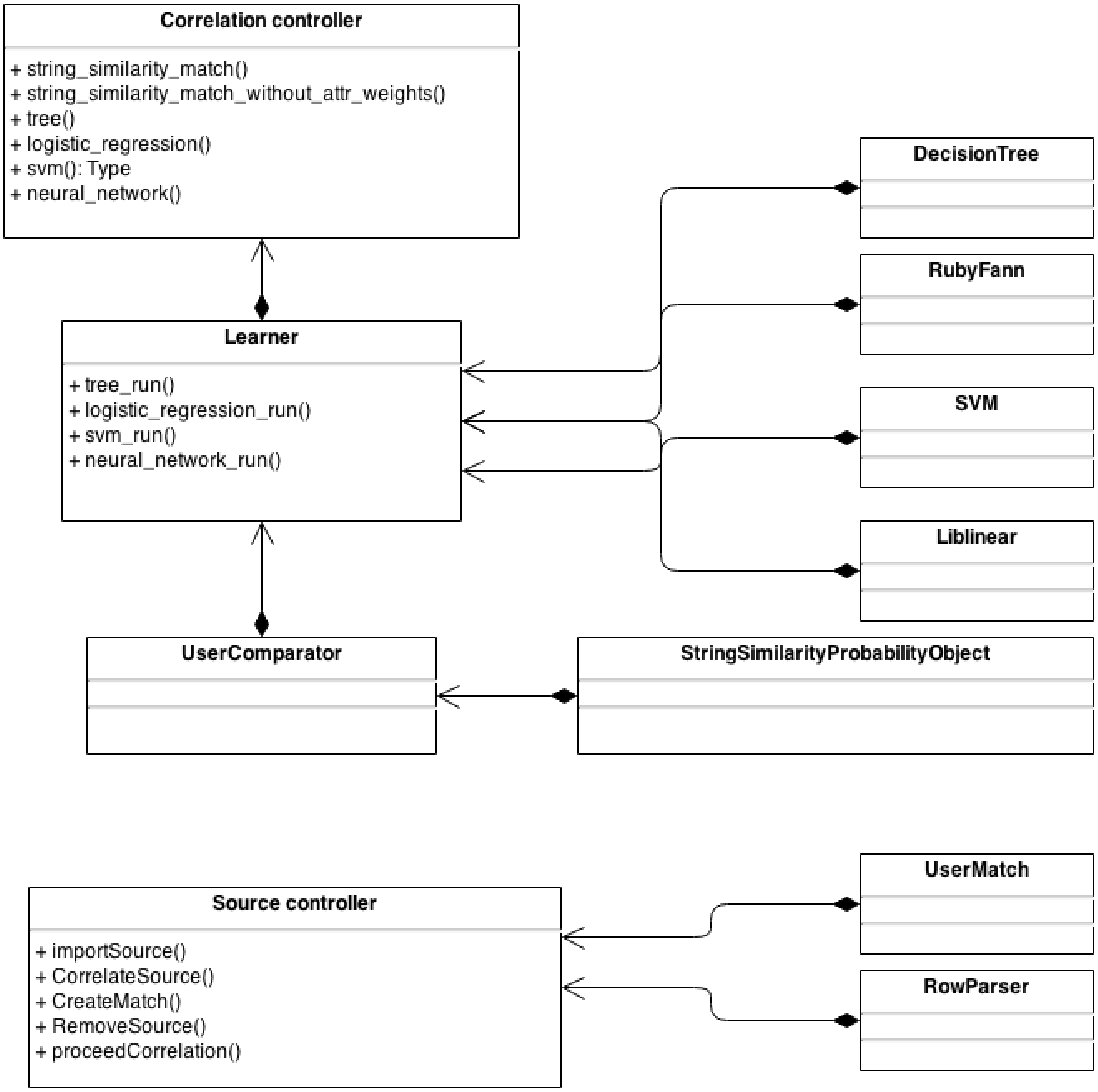
B Class diagram

Models



Class diagram of models in correlation framework shows relations between UserSource and UserObjects. UserObjectRaw has attributes, and definition of these attributes is stored in UserObjectMetadata. When the classifier is saved, the object ModelSerialized is created and model is stored into filesystem. Every UserSource object has SourceAttributeWeights which are used in creating similarity vector.

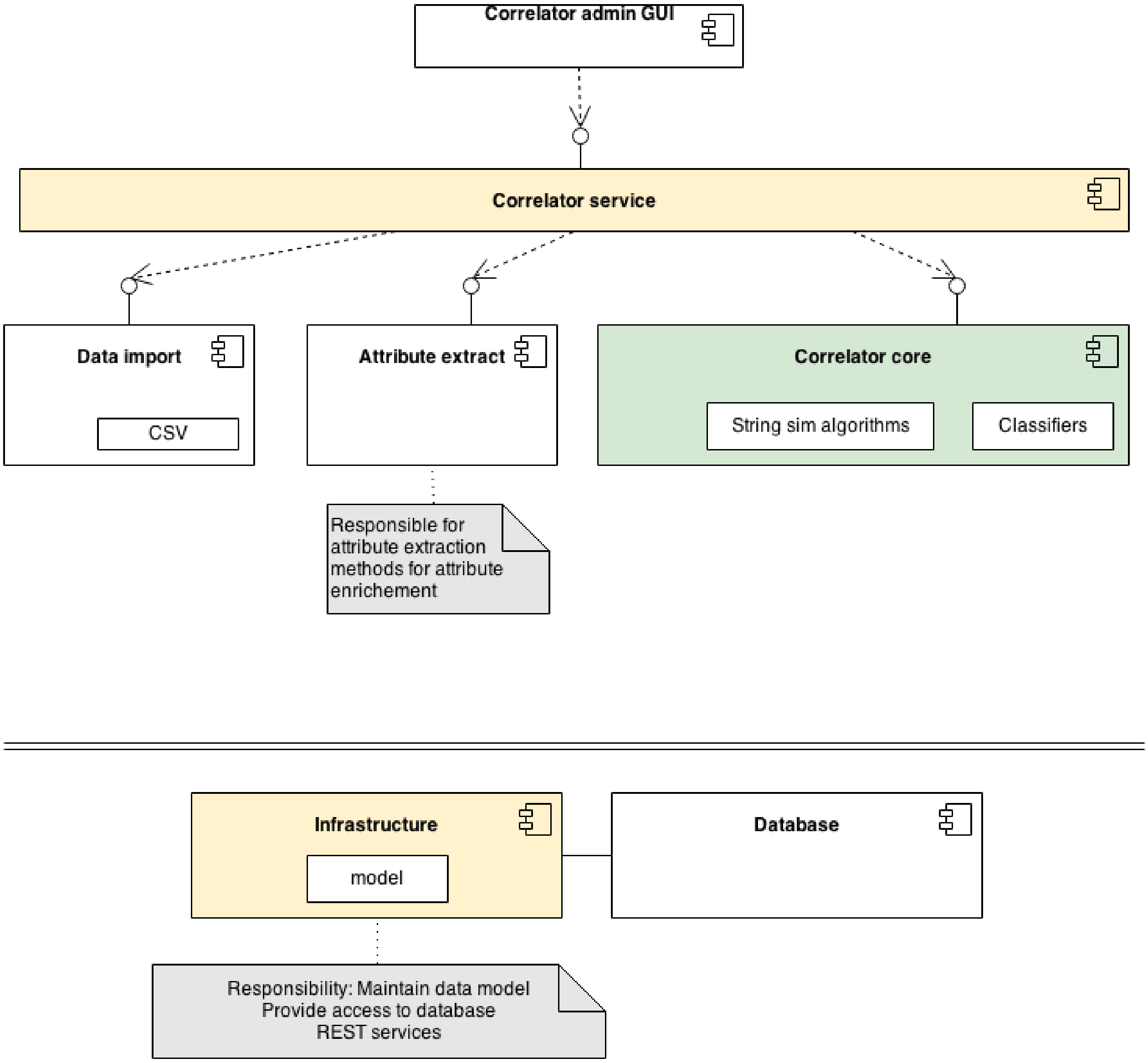
Class model



In this model, two main parts are correlation controller, which handles all machine learning algorithms and string similarity algorithms. Source controller handles importing, editing and removing sources. Parsing input is also part of source controller. In our application there are several software design patterns for example singleton pattern – concept of current user. Everytime the user is signed in, one current user object is created. Dependency injection pattern is also used as instances are passed to methods as arguments i.e. in create source method.

C Component diagram

On the component diagram below there are main components – Crrelator service, which aggregates data import, attribute extraction and correlation core component. Correlation core component handles string similarity algorithms and classifiers. Infrastructure component handles model and access to database. It also provides MVC in web based application, libraries for connection to database, logging infrastructure etc.



D User manual

Installation

Correlation framework is right now available as web application. For local installation you need:

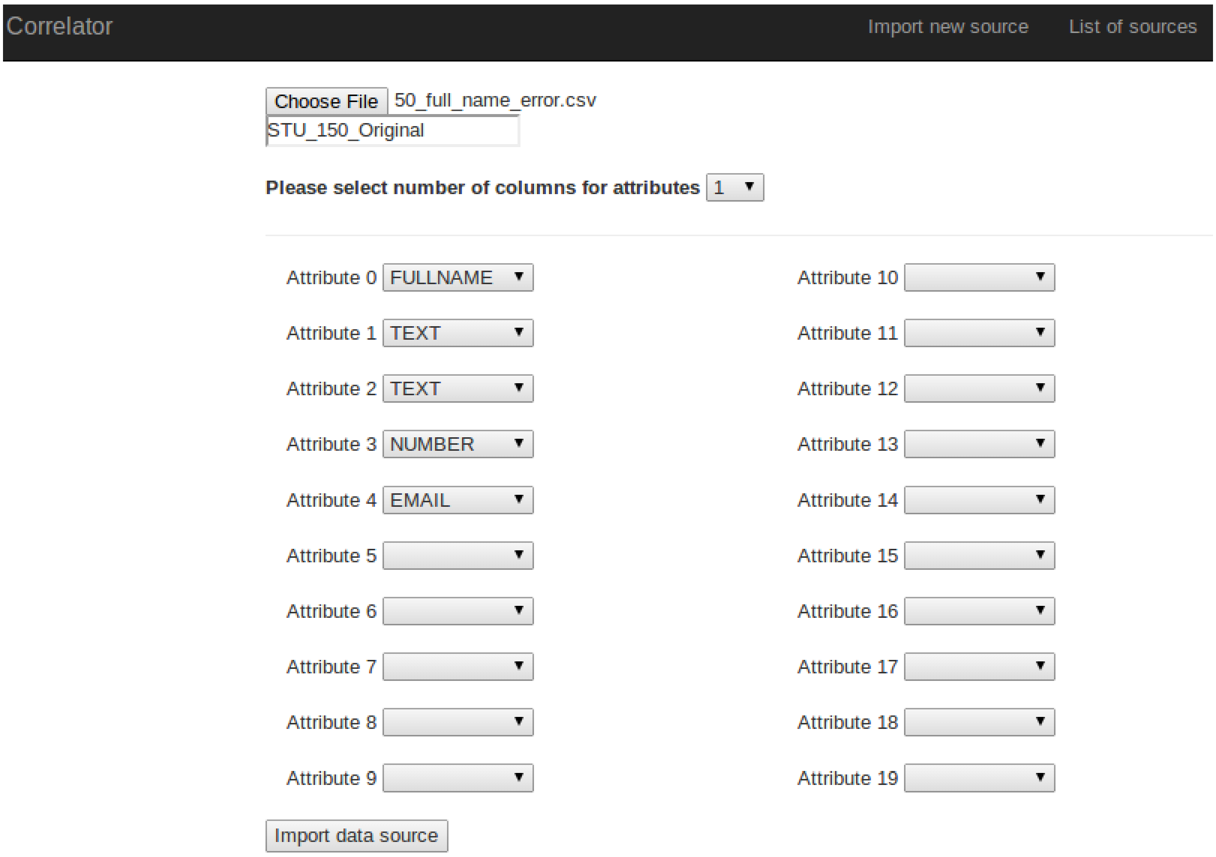
- Ruby on Rails version 4
- Ruby 2

Installation steps:

- Bundle install (Installation of libraries)
- Rake db:setup (Creation of database)
- Rails s (Start local server on localhost)

Import source

First step is to import data trough import source screen (Screen 1). User choose file in CSV format, choose number of attributes and choose type for each attribute. Type can be: fullname, text, number, email, given name, surname.



50_full_name_error.csv

STU_150_Original

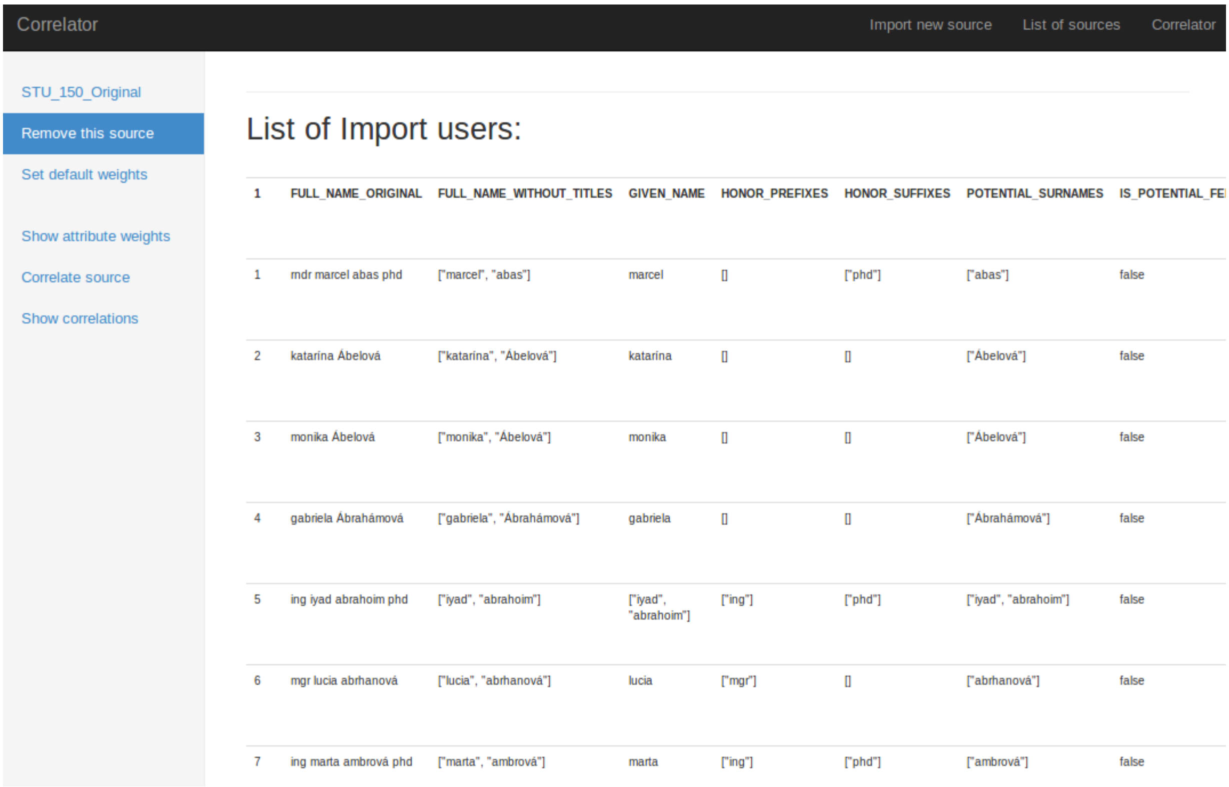
Please select number of columns for attributes 1

Attribute 0	FULLNAME	Attribute 10	
Attribute 1	TEXT	Attribute 11	
Attribute 2	TEXT	Attribute 12	
Attribute 3	NUMBER	Attribute 13	
Attribute 4	EMAIL	Attribute 14	
Attribute 5		Attribute 15	
Attribute 6		Attribute 16	
Attribute 7		Attribute 17	
Attribute 8		Attribute 18	
Attribute 9		Attribute 19	

Screen 1. Import source

Source detail

Afrer importing source data, user can see list of records with all attributes. He can set default attribute weights manually or automatically. He can correlate this data source with another source already inported.



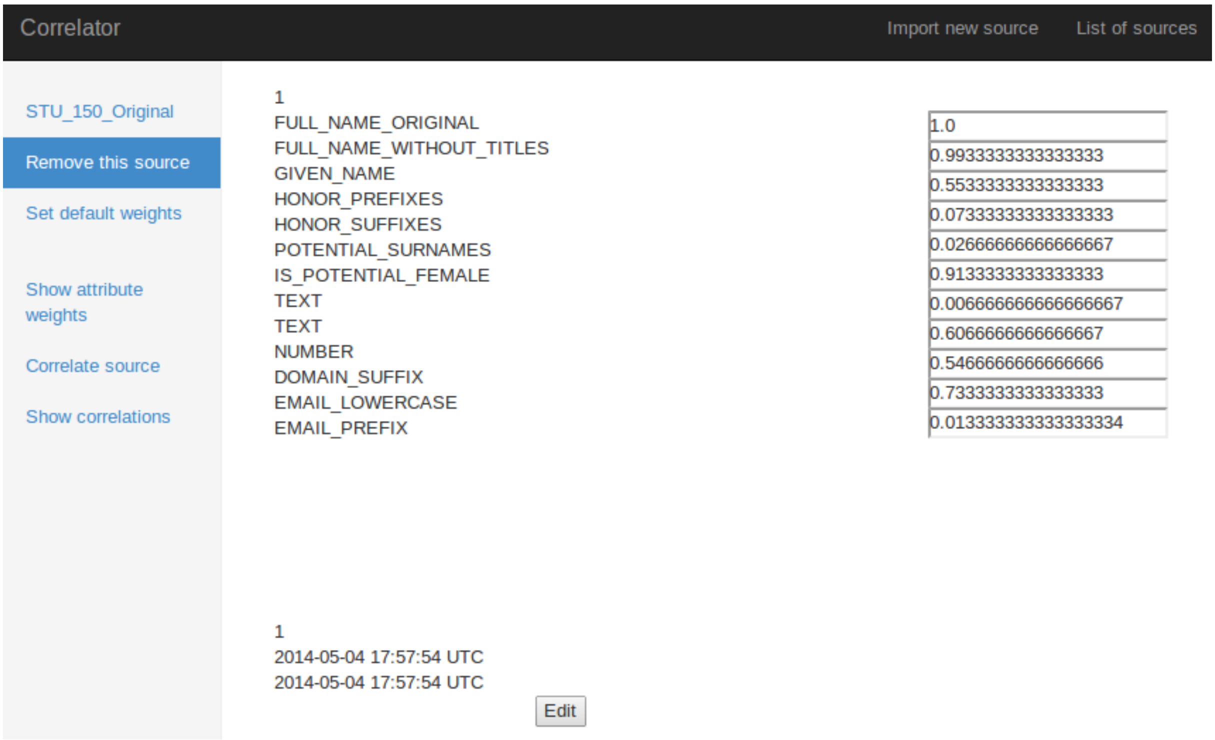
List of Import users:

1	FULL_NAME_ORIGINAL	FULL_NAME_WITHOUT_TITLES	GIVEN_NAME	HONOR_PREFIXES	HONOR_SUFFIXES	POTENTIAL_SURNAMES	IS_POTENTIAL_FE
1	rndr marcel abas phd	["marcel", "abas"]	marcel	[]	["phd"]	["abas"]	false
2	katarína Ábelová	["katarína", "Ábelová"]	katarína	[]	[]	["Ábelová"]	false
3	monika Ábelová	["monika", "Ábelová"]	monika	[]	[]	["Ábelová"]	false
4	gabriela Ábrahámová	["gabriela", "Ábrahámová"]	gabriela	[]	[]	["Ábrahámová"]	false
5	ing iyad abrahoim phd	["iyad", "abrahoim"]	["iyad", "abrahoim"]	["ing"]	["phd"]	["iyad", "abrahoim"]	false
6	mgr lucia abrhanová	["lucia", "abrhanová"]	lucia	["mgr"]	[]	["abrhanová"]	false
7	ing marta ambrová phd	["marta", "ambrová"]	marta	["ing"]	["phd"]	["ambrová"]	false

Screen 2. Source detail

Attribute weights edit screen

User can edit attribute weights for each attribute.



1
FULL_NAME_ORIGINAL
FULL_NAME_WITHOUT_TITLES
GIVEN_NAME
HONOR_PREFIXES
HONOR_SUFFIXES
POTENTIAL_SURNAMES
IS_POTENTIAL_FEMALE
TEXT
TEXT
NUMBER
DOMAIN_SUFFIX
EMAIL_LOWERCASE
EMAIL_PREFIX

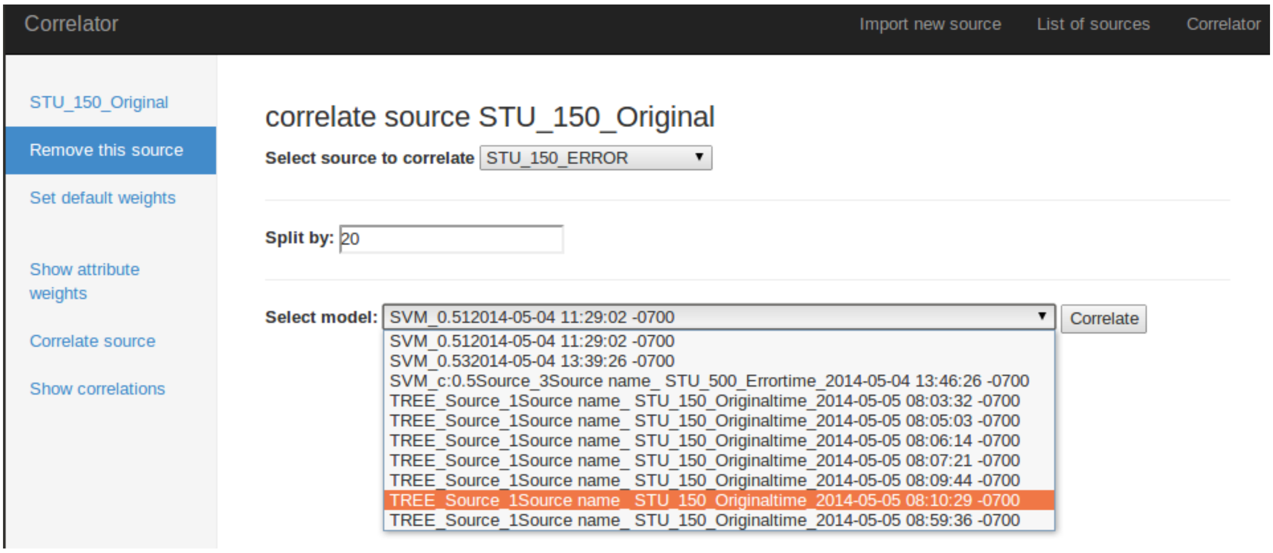
1.0
0.9933333333333333
0.5533333333333333
0.07333333333333333
0.02666666666666667
0.9133333333333333
0.006666666666666667
0.6066666666666667
0.5466666666666666
0.7333333333333333
0.013333333333333334

1
2014-05-04 17:57:54 UTC
2014-05-04 17:57:54 UTC

Screen 3. Edit attribute weights

Correlate source with another source

After user choose source to correlate with, he inputs partition size and choose appropriate model from list.



correlate source STU_150_Original

Select source to correlate STU_150_ERROR

Split by: 20

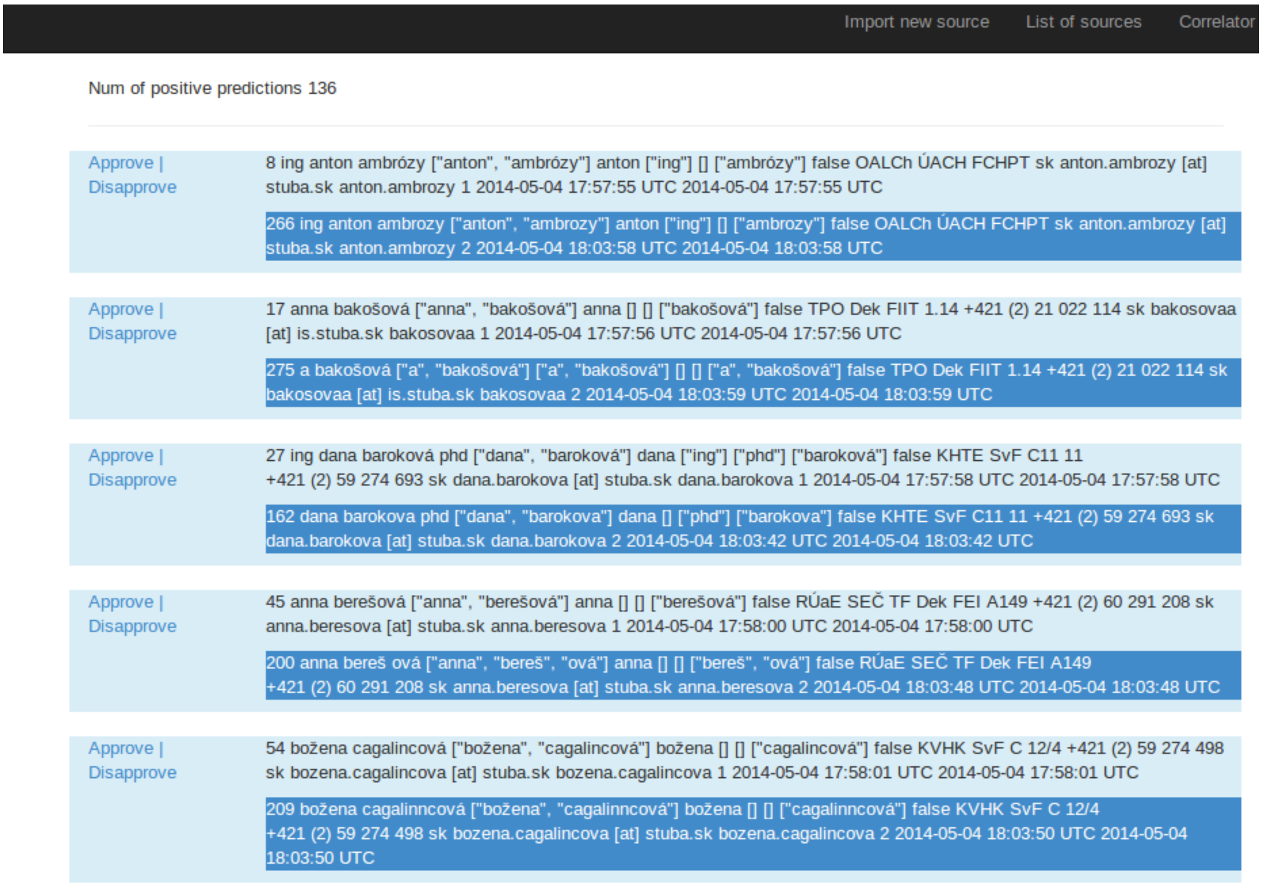
Select model: SVM_0.512014-05-04 11:29:02 -0700

SVM_0.512014-05-04 11:29:02 -0700
SVM_0.532014-05-04 13:39:26 -0700
SVM_c:0.5Source_3Source name_ STU_500_Errortime_2014-05-04 13:46:26 -0700
TREE_Source_1Source name_ STU_150_Originaltime_2014-05-05 08:03:32 -0700
TREE_Source_1Source name_ STU_150_Originaltime_2014-05-05 08:05:03 -0700
TREE_Source_1Source name_ STU_150_Originaltime_2014-05-05 08:06:14 -0700
TREE_Source_1Source name_ STU_150_Originaltime_2014-05-05 08:07:21 -0700
TREE_Source_1Source name_ STU_150_Originaltime_2014-05-05 08:09:44 -0700
TREE_Source_1Source name_ STU_150_Originaltime_2014-05-05 08:10:29 -0700
TREE_Source_1Source name_ STU_150_Originaltime_2014-05-05 08:59:36 -0700

Screen 4. Correlate source

Result of correlation process

User see list of records and can manually verify records.



Num of positive predictions 136

8 ing anton ambrózy ["anton", "ambrózy"] anton ["ing"] [] ["ambrózy"] false OALCh ÚACH FCHPT sk anton.ambrozy [at] stuba.sk anton.ambrozy 1 2014-05-04 17:57:55 UTC 2014-05-04 17:57:55 UTC

266 ing anton ambrozy ["anton", "ambrozy"] anton ["ing"] [] ["ambrozy"] false OALCh ÚACH FCHPT sk anton.ambrozy [at] stuba.sk anton.ambrozy 2 2014-05-04 18:03:58 UTC 2014-05-04 18:03:58 UTC

17 anna bakošová ["anna", "bakošová"] anna [] [] ["bakošová"] false TPO Dek FIIT 1.14 +421 (2) 21 022 114 sk bakosovaa [at] is.stuba.sk bakosovaa 1 2014-05-04 17:57:56 UTC 2014-05-04 17:57:56 UTC

275 a bakošová ["a", "bakošová"] ["a", "bakošová"] [] [] ["a", "bakošová"] false TPO Dek FIIT 1.14 +421 (2) 21 022 114 sk bakosovaa [at] is.stuba.sk bakosovaa 2 2014-05-04 18:03:59 UTC 2014-05-04 18:03:59 UTC

27 ing dana baroková phd ["dana", "baroková"] dana ["ing"] ["phd"] ["baroková"] false KHTE SvF C11 11 +421 (2) 59 274 693 sk dana.barokova [at] stuba.sk dana.barokova 1 2014-05-04 17:57:58 UTC 2014-05-04 17:57:58 UTC

162 dana barokova phd ["dana", "barokova"] dana [] ["phd"] ["barokova"] false KHTE SvF C11 11 +421 (2) 59 274 693 sk dana.barokova [at] stuba.sk dana.barokova 2 2014-05-04 18:03:42 UTC 2014-05-04 18:03:42 UTC

45 anna berešová ["anna", "berešová"] anna [] [] ["berešová"] false RÚaE SEČ TF Dek FEI A149 +421 (2) 60 291 208 sk anna.beresova [at] stuba.sk anna.beresova 1 2014-05-04 17:58:00 UTC 2014-05-04 17:58:00 UTC

200 anna bereš ová ["anna", "bereš", "ová"] anna [] [] ["bereš", "ová"] false RÚaE SEČ TF Dek FEI A149 +421 (2) 60 291 208 sk anna.beresova [at] stuba.sk anna.beresova 2 2014-05-04 18:03:48 UTC 2014-05-04 18:03:48 UTC

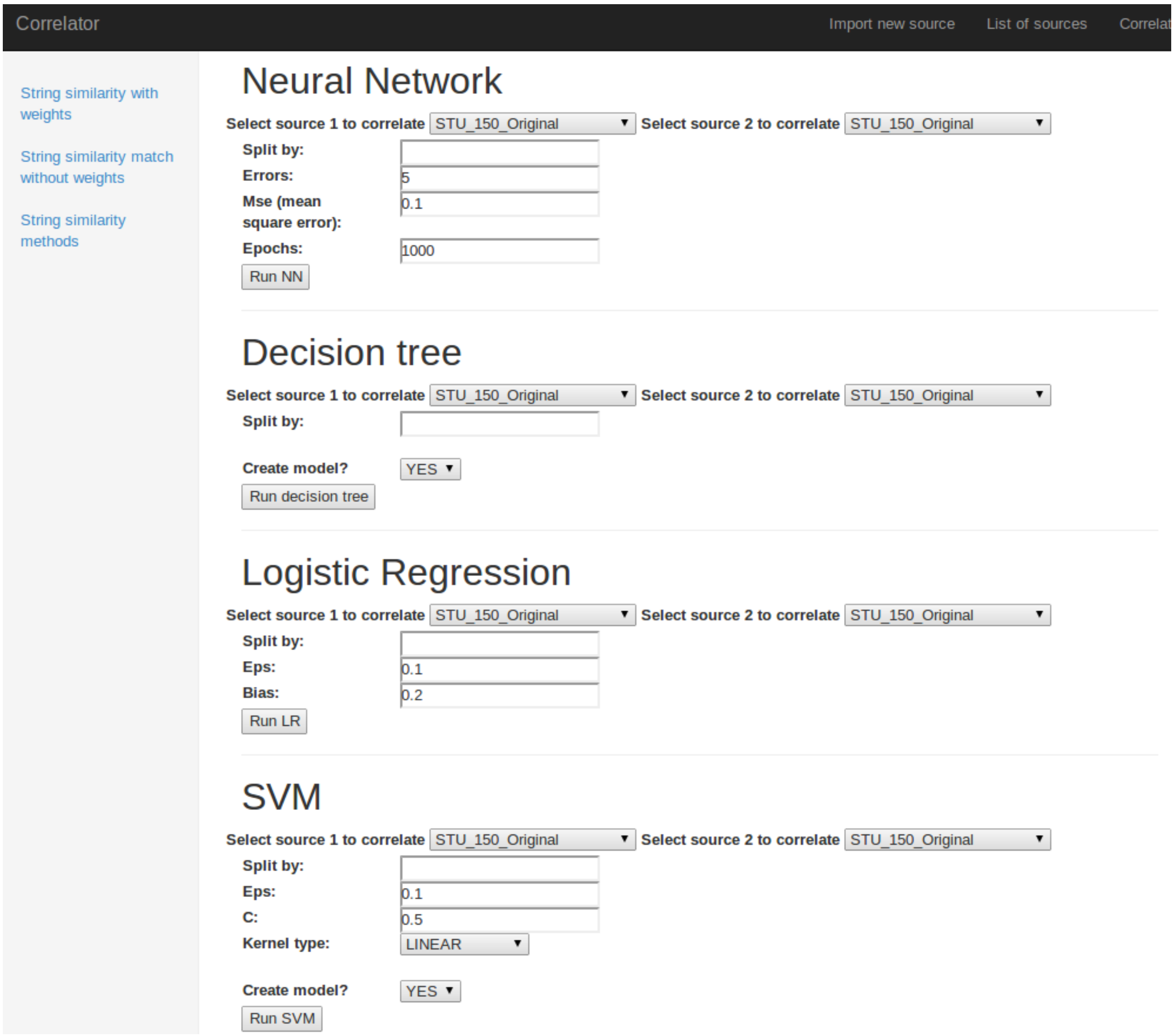
54 božena cagalincová ["božena", "cagalincová"] božena [] [] ["cagalincová"] false KVHK SvF C 12/4 +421 (2) 59 274 498 sk bozena.cagalincova [at] stuba.sk bozena.cagalincova 1 2014-05-04 17:58:01 UTC 2014-05-04 17:58:01 UTC

209 božena cagalinncová ["božena", "cagalinncová"] božena [] [] ["cagalinncová"] false KVHK SvF C 12/4 +421 (2) 59 274 498 sk bozena.cagalincova [at] stuba.sk bozena.cagalincova 2 2014-05-04 18:03:50 UTC 2014-05-04 18:03:50 UTC

Screen 5. Results of correlation process

Correlator machine learning model setup screen

User can train own model for each data source. He can choose partition size, parameters for each classifier and he can save model.


Correlator
Import new source
List of sources
Correlat
String similarity with weights
String similarity match without weights
String similarity methods

Neural Network

Select source 1 to correlate: STU_150_Original
Select source 2 to correlate: STU_150_Original
Split by:
Errors: 5
Mse (mean square error): 0.1
Epochs: 1000

Decision tree

Select source 1 to correlate: STU_150_Original
Select source 2 to correlate: STU_150_Original
Split by:
Create model? YES

Logistic Regression

Select source 1 to correlate: STU_150_Original
Select source 2 to correlate: STU_150_Original
Split by:
Eps: 0.1
Bias: 0.2

SVM

Select source 1 to correlate: STU_150_Original
Select source 2 to correlate: STU_150_Original
Split by:
Eps: 0.1
C: 0.5
Kernel type: LINEAR
Create model? YES

Screen 6. Source detail

E Experiment examples

Results of comparing 150 original records with 150 error records.

Partition size (%)	Neural Network	Logistic Regression	SVM	Decision Tree
20	0,14	0,89	0,9	0,96
40	0,21	0,911	0,96	0,98
60	0,19	0,9	0,97	0,97
• Results are F1 score				

Results of comparing 500 original records with 500 records containing 150 error records.

Partition size (%)	Neural Network	Logistic Regression	SVM	Decision Tree
20	0,12	0,84	0,92	0,93
40	0,2	0,85	0,9	0,92
60	0,21	0,86	0,91	0,92
• Results are F1 score				

Time complexity of algorithms in addition to partitioning size (comparing 150 records vs. 150 records):

Partition size (%)	Neural Network	Logistic Regression	SVM	Decision Tree
20	30	6	13	130
40	45	35	28	490
60	68	42	50	930
• Results are shown in seconds				

Decision tree model trained rule-set example

Decision Tree – Trained rule set with F1 score 0.962 (150 user records)

```
FULL_NAME_ORIGINAL >= 0.9787452210343168
=> 1 ()

FULL_NAME_ORIGINAL < 0.9787452210343168
EMAIL_LOWERCASE >= 0.8201058201058201
TEXT >= 0.22334299516908213
=> 0 ()

FULL_NAME_ORIGINAL < 0.9787452210343168
EMAIL_LOWERCASE >= 0.8201058201058201
TEXT < 0.22334299516908213
=> 1 ()

FULL_NAME_ORIGINAL < 0.9787452210343168
EMAIL_LOWERCASE >= 0.8201058201058201
TEXT < 0.22334299516908213
IS_POTENTIAL_FEMALE >= 0.4150877192982456
=> 1 ()

FULL_NAME_ORIGINAL < 0.9787452210343168
EMAIL_LOWERCASE >= 0.8201058201058201
TEXT < 0.22334299516908213
IS_POTENTIAL_FEMALE < 0.4150877192982456
=> 0 ()

FULL_NAME_ORIGINAL < 0.9787452210343168
EMAIL_LOWERCASE < 0.8201058201058201
=> 0 ()
```

F Content of electronic medium

Electronic medium has this structure:

/ Guide
/ Prototype – implemented prototype
/ Thesis – electronic version of diploma thesis

G Resume

Úvod

Systémy na správu digitálnych identít riadia dátové úložiská identít, integrujú ich a poskytujú prístup k rôznym systémom. Dátové úložiská identít obsahujú záznamy s údajmi o digitálnych identitách, ktoré môžu obsahovať nekonzistentné dáta. V súčasnosti existujú rôzne systémy na správu identít ktoré pracujú s niekoľkými desiatkami až stovkami dátových úložísk. Čím viac systémov je integrovaných, tým náročnejší je proces mapovania a korelácie údajov o identitách. Hlavným problémom je nutnosť manuálneho mapovania, ktoré je v mnohých prípadoch príliš náročné na čas. Existujúce riešenia v oblasti automatickej korelácie údajov o identitách ako napríklad korelačné pravidlá nie sú dostačujúce. Našim cieľom je preto navrhnúť a implementovať efektívnu metódu na automatickú koreláciu údajov o digitálnych identitách.

Manažment identít

Manažment identít riadi identity v digitálnom priestore. Je to kombinácia technológií a postupov na reprezentovanie a rozpoznávanie entít ako digitálnych entít v digitálnom priestore. Každá organizácia má iné nároky na manažment identít, a tak je nutné individuálne prispôsobovanie a nastavovanie procesov v rámci manažmentu identít. Hlavnou úlohou systémov na správu a riadenie identít je integrácia údajov o identitách, spracovanie a riadenia životného cyklu digitálnych identít – vytvorenie, úprava, zrušenie digitálnej identity. Digitálna identita obsahuje údaje o osobe, reprezentované pomocou technických prostriedkov ako množinu atribútov popisujúcich danú osobu. Používateľský účet je entita obsahujúca informácie o osobe a kontexte v ktorom bol účet vytvorený. Môže obsahovať napríklad osobné údaje, prístupové práva a systémové nastavenia.

V súčasnej dobe je oblasť manažmentu identít automatizovaný proces vzhľadom na objemy dát s ktorými musí pracovať. Manažment identít je dôležitý pre akúkoľvek organizáciu, ktorá chce poskytovať práva a prístupy do podsystémov pre svojich zamestnancov a zákazníkov. Systémy na manažment identít pozostáva z troch hlavných technologických častí:

- Úložiská identít
- Riadenie prístupov
- Správa účtov (angl. „provisioning“)

Úložiská identít obsahujú informácie o používateľských účtoch a sú často zdieľané rôznymi aplikáciami v rámci organizácie, ale aj mimo nej. Úložiská identít používajú rôzne technológie na správu údajov napríklad LDAP (Lightweight Directory Access Protocol).

Riadenie prístupu je oblasť, ktorá sa zaoberá autentifikáciou a autorizáciou v rámci systému. Bezpečnosť aplikácií je väčšinou hlavnou požiadavkou a tak je nutné chrániť citlivé dáta o identitách. Autentifikácia je proces verifikácie používateľa – overenie či je osoba naozaj tá za ktorú sa vydáva. Využívajú sa na to overovanie verejných kľúčov, prihlasovacích mien a hesiel. Autorizácia je proces, ktorý nasleduje po autentifikácii a má zaručiť prístup k systémovým (aplikačným) zdrojom. Tieto zdroje môžu byť služby, funkcionalita systému, prístup k údajom a iné.

V rámci riadenia prístupov v systémoch na správu identít sa využíva koncept „Single sign-on angl.“ jednotné prihlásenie, ktorý zabezpečuje prístup k viacerým systémom bez nutnosti opätovného prihlasovania.

Provisioning je časť manažmentu identít, ktorý sa zaoberá riadením a integrovaním úložísk identít. Zabezpečuje synchronizáciu dát z viacerých zdrojov. Rovnako riadi mechanizmus vytvárania, modifikovania a rušenia používateľských účtov a prístupov k systémovým zdrojom. Napríklad v prípade vytvorenia používateľského účtu v jednom systéme je nutné vytvoriť účty vo všetkých ostatných systémoch.

Zlučovanie v rámci manažmentu identít je proces synchronizácie viacerých úložísk identít a dátových zdrojov s cieľom poskytnúť centrálny mechanizmus správy identít na jednom mieste. V rámci zlučovania sa určuje zhodnosť záznamov o identitách z rôznych systémov.

Korelácia identít

Používateľ môže v rámci organizácie pristupovať k rôznym systémom s rôznymi používateľskými účtami. Tieto účty môžu obsahovať rôzne atribúty ako napríklad prihlasovacie meno („tjendek, tomas.jendek, t.jendek“) a iné. Systém na správu identít musí pracovať s rôznymi používateľskými účtami pre jednu identitu a tak udržovať spojenia medzi účtami. V prípade, ak si používateľ v jednom systéme zmení heslo, musí sa táto zmena prešíriť aj do iných systémov. Korelácia identít je proces spájania a validácie záznamov používateľských účtov, ktoré spája vlastníctvo jednej identity. Samotné spájanie je realizované pomocou porovnávania hodnôt atribútov používateľského účtu. Na to sa využívajú zväčša korelačné pravidlá, ktoré spájajú záznamy na základe zhody určeného atribútu. Takáto korelácia nie je dostačujúca a je nutné manuálne spájanie záznamov v komplexnejších prípadoch.

Spájanie záznamov je proces pri ktorom sa na základe porovnávania hodnôt atribútov určí podobnosť záznamov a následne sú určené potenciálne zhody. Existujú dva prístupy spájania záznamov:

- pravdepodobnostné
- deterministické

Deterministické spájanie sa inak nazýva „presné“ kvôli spájaniu záznamov ktorých hodnoty atribútov sa úplne zhodujú. Pravdepodobnostné spájanie je založené na porovnávaní rôznych hodnôt atribútov a ich váh. Podobnosť záznamov je pomocou spájacích kritérií vyhodnotená ako zhoda, alebo nezhoda. Spájanie záznamov je komplexný proces ktorý sa skladá z:

- čistenia a štandardizácie dát

- porovnávania
- klasifikácie
- vyhodnotenia

V prvom kroku je zjednotený formát dát, sú odstránené nadbytočné medzery. Následne sú záznamy porovnávané pomocou algoritmov na určovanie podobnosti reťazcov. Po určení podobnosti nastáva klasifikácia porovnávaných záznamov a vyhodnotenie – overenie výsledkov.

Algoritmy na hľadanie podobnosti reťazcov sú:

- znakové
- vektorové

Znakové algoritmy a techniky sú založené na modifikačné operácie ako napríklad odstránenie znaku, výmena znaku a ich rátanie. Znakové algoritmy sú vhodné na určovanie podobnosti reťazcov v prípadoch, keď obsahujú preklepy, alebo skratky. Pre dlhšie reťazce sú znakové algoritmy neefektívne. Príkladom znakových algoritmov sú:

- Levenshteinova vzdialenosť
- Damerau-Levenshteinova vzdialenosť
- Bag vzdialenosť
- Smith-Watermant
- Jaro vzdialenosť
- N-gramy

Vektorové techniky sú určené na dlhšie reťazce, pretože reťazce reprezentujú ako tokeny ktorých poradie nie je dôležité. Reťazce sú reprezentované ako riedke n-dimenzionálne vektory reálnych čísel, kde každá hodnota patrí tokenu v rámci reťazca. Príkladom je napríklad TF-IDF metóda.

Strojové učenie

Strojové učenie je oblasť umelej inteligencie ktorá umožňuje počítačovému systému „učiť sa“ - zlepšovať výsledky na základe predchádzajúcich výsledkov. Strojové učenie objavuje pravidelnosti a klasifikuje dáta. Klasifikácia je založená na trénovacej množine s označenými triedami a testovacej, na ktorú sa uplatňuje naučený model. Algoritmy strojového učenia sú napríklad:

- K najbližších susedov
- Podporné vektory (SVM)
- Neurónové siete
- Rozhodovacie stromy
- Logistická regresia

Návrh metódy

Nami navrhovaný metóda je založená na algoritmoch podobnosti reťazcov, ktoré určujú podobnosť medzi hodnotami atribútov porovnávaných záznamov a algoritmoch strojového učenia pre automatickú koreláciu. Metóda sa skladá zo základných krokov:

- Príprava dát, normalizácia atribútov a extrakcia atribútov
- Rozdeľovanie záznamov, nastavenie základných váh pre atribúty
- Aplikácia vhodných algoritmov na určovanie podobnosti reťazcov
- Aplikácia klasifikátorov – trénovanie modelu a klasifikácia
- Verifikácia výsledkov

Príprava dát je prvý krok, pri ktorom sa vstupné dáta ukladajú v jednotnom formáte do databázy. Okrem toho sa jednotlivé atribúty transformujú a štandardizujú. V našej práci navrhujeme import CSV formátu („comma separated value" ang.) kde záznamy vyzerajú napr."id: first_attribute_value; second_attribute_value". Normalizácia vstupných dát spočíva v zmene všetkých znakov na malé písmená, odstránenie interpunkčných znamienok, nahradenie viacero medzier jednou medzerou a odstránenie medzier pred a za reťazcom.

Vstupné dáta obsahujú rôzne atribúty , niektoré z nich môžu byť rozdelené na pod-atribúty, ako napríklad atribút meno môže byť rozdelený na krstné meno, tituly a priezvisko. Navrhujeme metódy na extrakciu pre atribúty krstné meno, priezvisko, tituly a emailovú adresu.

Určovanie korelácie pri veľkom množstve záznamov v úložiskách identít môže byť neefektívne vzhľadom na to, že pri porovnávaní záznamov (karteziánsky súčin) dochádza k porovnávaniu každého záznamu s každým. V našej metóde navrhujeme rozdeľovací mechanizmus vďaka ktorému sa na základe frekvencie výskytu atribútu zoradia záznamy a rozdelia sa na rovnaké skupiny. Tieto skupiny sú navzájom porovnávané. Napríklad 6000 záznamov zoradíme podľa atribútu „krstné meno" a určíme veľkosť skupiny na 100. Vznikne tak 60 skupín, ktoré sú porovnávané tak, že 1. skupina z prvého zdroja identít je porovnávaná s 1. skupinou z druhého zdroja.

Korelácia záznamov je založená na podobnosti hodnôt atribútov. V práci využívame funkciu $sim_a(attr1, attr2)$, ktorá pre každú dvojicu atribútov z dvoch záznamov určí podobnosť. Všetky hodnoty podobnosti atribútov sú uložené vo forme vektora [$sim_{a,}$ $sim_{b,}$ sim_c ...]. Každý vektor obsahuje hodnoty od 0 po 1, kde 1 je zhoda a 0 je nezhoda. Analyzovali sme rôzne algoritmy na určovanie podobnosti a pre jednotlivé atribúty sme určili jeden algoritmus, ktorý dosahoval najlepšie výsledky. Pre krstné meno je to Jaro-Winkler algoritmus, pre priezvisko je to Jaro vzdialenosť atď. Po vytvorení vektora s podobnosťami záznamov prichádza strojové učenie, pomocou ktorého algoritmov sa natrénuje model, ktorý bude aplikovateľný na nové dáta a klasifikuje porovnávané záznamy ako zhodu, alebo nezhodu. Proces klasifikácie pozostáva z:

- Rozdelenie záznamov do skupín
- Vytvorenie vektora podobností
- Vytvorenia trénovacej množiny

- Aplikovanie modelu na testovaciu množinu

Implementácia metódy

V našej práci sme navrhli a implementovali korelačný rámec na koreláciu identít z rôznych zdrojov. Cieľom rámca je spraviť tento proces čo najviac automatický. Scenár pre použitie:

- Výber vstupných dát (zdroje údajov o identitách)
- Import dát vo formáte CSV, nastavenie váh atribútov (manuálne/automatické)
- Záznamy sú automaticky zoradené a rozdelené do skupín
- Výpočet podobnostných vektorov
- Trénovanie/aplikácia modelu
- Verifikácia výsledkov

Vstupné dáta sú spracované a normalizované. Vytvoria sa objekty pre reprezentáciu identity. Pre každý objekt sú vypočítané váhy pre jednotlivé atribúty. Objekty sú rozdelené do skupín pre následné určovanie podobnosti. Vo fáze vytvárania modelu sú vektory obohatené o ručne určenú triedu (či vektor označuje zhodné záznamy, alebo nie). Používateľ vyberie klasifikátor, nastaví parametre a spustí trénovanie modelu. Vytvorený model následne aplikuje na testovaciu množinu a verifikuje výsledky. V prípade, že je model už vytvorený, aplikuje ho na dáta.

V práci sme využili dáta o zamestnancoch Slovenskej technickej univerzity v Bratislave zo zdrojov AIS (Akademický informačný systém). Rovnaké dáta sme získali aj z webu www.portalvs.sk (portál vysokých škôl). Dáta obsahovali údaje o zamestnancoch fo formáte: kompletné meno, organizačná jednotka, číslo kancelárie, telefón, email ("*Ing. Marta Ambrová, PhD.;OAT ÚATM FCHPT;SB172;+421 (2) 59 325 783;marta.ambrova [at] stuba.sk*").

Dáta obsahovali 6625 záznamov a pri experimentoch sme vytvorili rôzne podmnožiny (50, 100, 150, 200, 500) záznamov, pre ktoré sme vytvorili umelé chyby – preklepy, vynechania slov, vynechania znakov, zmenu pozície slov atď. Implementácia korelačného rámca je realizovaná ako webová služba s využitím Ruby on Rails webového programového rámca a databázy PostgreSQL.

Experimenty

Na vyhodnotenie výsledkov klasifikácie využívame metriky presnosť, pokrytie a F1. V prvom experimente sme sa zamerali na určenie preddefinovaných váh atribútov. Zisťovali sme rozlišovaciu schopnosť daného atribútu a podľa toho sme nastavili preddefinované váhy atribútov. Tie sú vypočítané ako súčet unikátnych výskytov hodnoty atribútu vydelený počtom záznamov. Na overenie vplyvu automaticky preddefinovaných váh atribútov sme vyskúšali určovať podobnosť na podmnožine 150 záznamov. Z výsledkov je vidno, že pri použití preddefinovaných atribútov sa zlepšilo F1 skóre o 4%.

Algoritmy na určovanie podobnosti záznamov boli jednotlivo vyskúšané na každý atribút a najlepšie výsledky zaznamenali:

- plné meno – Jaro-Winkler vzdialenosť
- krstné meno – Jaro-Winkler vzdialenosť
- plné meno s titulmi – Jaro-Winkler vzdialenosť
- tituly pred menom – Levenshtein vzdialenosť
- tituly za menom – Levenshtein vzdialenosť
- potenciálne priezviská – Jaro
- TEXT – Ngram (N = 2)
- Čísla – Jaro
- doménová prípona – Jaro-Winkler vzdialenosť
- email malým písmom – Levenshtein vzdialenosť
- emailový prefix – Jaro-Winkler vzdialenosť

Vykonali sme experimenty s rôznymi algoritmami strojového učenia a rôznym nastavenám parametrov pre tieto algoritmy. V prvom experimente sme porovnávali algoritmus K najbližších susedov s algoritmom podporných vektorov s využitím preddefinovaných váh atribútov a bez rozdeľovania záznamov. Podporné vektory zaznamenali o 3% vyššie F1 skóre. Ďalší experiment porovnával algoritmus podporných vektorov, neurónových sietí a logistickej regresie. Experiment sme realizovali s využitím rozdeľovania a porovnali sme ho aj bez využitia rozdeľovania. Výsledky ukázali, že najlepšie klasifikoval algoritmus logistickej regresie s rozdeľovaním. Pre neurónovú sieť sa nám nepodarilo nastaviť parametre a neuróny v skrytej vrstve tak, aby sme dosiahli porovnateľné výsledky s ostatnými algoritmami.

Pri práci s veľkým množstvom záznamov (6000) prichádza k porovnaniu 36 000 000-krát v prípade, že nevyužijeme rozdeľovanie. To má zásadný vplyv na dĺžku a kvalitu klasifikačného modelu. V našom experimente sme testovali rôzne veľké skupiny a sledovali sme vplyv na výsledok a dĺžku klasifikácie. Výsledky ukázali, že príliš malé skupiny sú síce rýchlo klasifikované, avšak presnosť je nízka. Naopak veľké skupiny sú časovo náročnejšie, avšak presnosť je dobrá. Najlepšiu presnosť sme zaznamenali pri veľkosti skupiny 40-50 %.

Zhrnutie

V našej práci sme analyzovali problematiku systémov pre správu identít s dôrazom na prístupy, ktoré spravujú životný cyklus identity a riadenie identít v rámci organizácie. Zamerali sme sa na oblasť automatickej korelácie záznamov. Zistili sme, že existujúce prístupy v rámci systémov na správu identít neposkytujú automatické korelačné nástroje, a tak je nutná manuálna práca. Hlavným cieľom našej práce je navrhnúť a implementovať automatickú korelačnú metódu. Naša metóda spracúva vstupné údaje o identitách, normalizuje záznamy a vytvára navyše odvodené atribúty z pôvodných pre lepšie určovanie podobnosti medzi záznamami. Naša metóda používa rôzne algoritmy na určovanie podobnosti reťazcov a aplikuje ich na konkrétne atribúty. Navrhujeme tiež mechanizmus na určovanie preddefinovaných váh atribútov na zlepšenie určovania podobnosti atribútov. Pomocou algoritmov strojového učenia – podporné vektory, logistická regresia a rozhodovacie stromy vytvárame klasifikačné modely a automaticky klasifikujeme záznamy o identitách. Naša metóda je súčasť korelačného rámca ktorý spracúva ako vstup CSV formát, poskytuje možnosť vytvorenia modelu pre jednotlivé úložiská identít a následne aplikáciu modelu pri klasifikovaní záznamov.