Complex Systems Informatics and Modeling Quarterly
CSIMQ, Issue 8, September/October 2016, Pages 81–96
Published online by RTU Press, https://csimq-journals.rtu.lv
https://doi.org/10.7250/csimq.2016-8.07
ISSN: 2255-9922 online

Employing Ontology-Alignment and Locality-Sensitive Hashing to Improve Attribute Interoperability in Federated eID Systems

Walter Priesnitz Filho[1], Carlos Ribeiro[1] and Thomas Zefferer[2]

[1]Instituto Superior Técnico, Universidade de Lisboa, Rovisco Pais 1, Lisboa, Portugal
[2]Institute for Applied Information Processing and Communications, Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria

{walter.filho, carlos.ribeiro}@tecnico.ulisboa.pt, thomas.zefferer@iaik.tugraz.at

Abstract. Achieving interoperability, i.e. creating identity federations between different Electronic identities (eID) systems, has gained relevance throughout the past years. A serious problem of identity federations is the missing harmonization between various attribute providers (APs). In closed eID systems, ontologies allow a higher degree of automation in the process of aligning and aggregating attributes from different APs. This approach does not work for identity federations, as each eID system uses its own ontology to represent its attributes. Furthermore, providing attributes to intermediate entities required to align and aggregate attributes potentially violates privacy rules. To tackle these problems, we propose the use of combined ontology-alignment (OA) approaches and locality-sensitive hashing (LSH) functions. We assess existing implementations of these concepts defining and using criteria that are special for identity federations. Obtained results confirm that proper implementations of these concepts exist and that they can be used to achieve interoperability between eID systems on attribute level. A prototype is implemented showing that combining the two assessment winners (AlignAPI for ontology-alignment and Nilsimsa for LSH functions) achieves interoperability between eID systems. In addition, the improvement obtained in the alignment process by combining the two assessment winners does not impact negatively the privacy of the user's data, since no clear-text data is exchanged in the alignment process.

Keywords: Interoperability, ontologies, ontology alignment, LSH functions, privacy.

1 Introduction

Electronic identities (eID) have become a critical concept of electronic services from both the private and the public sector. For instance, e-government solutions use eIDs to identify and authenticate citizens in online governmental processes. In most cases, an eID is a unique number that is assigned to a user and unambiguously identifies this user. A typical identification/authentication process involves several entities. The Identity Provider (IdP) establishes, maintains, and secures the electronic identity associated with a user, and verifies the identity of that user. The Relying Party (RP) makes transaction decisions based on receipt, validation, and acceptance of a user's authenticated credentials and attributes within the Identity System (IS). For instance, a Service Provider (SP), which, e.g. provides an e-government service, can assume the role of the relying party.

In most identity systems, identity attributes are used together with the eID of a user. For instance, during an identification/authentication process, the relying party might additionally be provided with the user's name and date of birth. From a conceptual perspective, identity attributes are provided by Attribute Providers (APs). In practice, identity provider and attribute provider can also be represented by one and the same entity.

The successful application of electronic identities requires that user, identity provider, and relying party are part of the same identity system. Recalling the e-government example, eIDs assigned to citizens by a national government can only be used to identify and authenticate at relying parties from the same country. As this is a significant limitation, several attempts have been made during the past years to achieve interoperability between different identity systems, i.e. to establish an eID federation. In Europe, the eIDAS Regulation[1] and the EU large-scale pilots STORK[2] and STORK 2.0[3] have, for instance, successfully established eID interoperability between EU member states (MS). As a result, citizens from MS_A can use their national eID to identify and authenticate at service providers from MS_B and vice versa.

While there have been significant advances in eID interoperability and in creating eID federations in the past years, the role of eID attributes has often been neglected. In most cases, interoperability of eID attributes is implicitly assumed. For instance, interoperability solutions developed by STORK and STORK 2.0 rely on a common set of attributes shared by all participating countries. Unfortunately, this is an oversimplification that does often not reflect reality. In practice, there is usually no harmonization between attribute providers of different identity systems. This lack of harmonization raises problems, if attribute providers from different identity systems store attributes of one and the same user. In this case, attributes from different identity systems potentially need to be aligned and aggregated during identification/authentication processes.

Within a single identity system, the problem of aligning and aggregating attributes from different attribute providers can be tackled by defining a common ontology for attributes. In scenarios that involve multiple identity systems and hence require attribute interoperability, this approach does not work for two reasons. First, each identity system typically uses its own specific ontology to represent eID attributes, which prevents a direct aligning or aggregation of attributes. Second, aligning and aggregating attributes requires an intermediate entity that performs these tasks. Exposing attributes to this entity might violate defined privacy rules.

To tackle these issues and to allow the aligning and aggregation of attributes in eID federations, new ground has to be broken. In this paper, we, therefore, propose the use of ontology-alignment (OA) solutions and locality-sensitive hashing (LSH) functions. We show how these two technologies can be used for the concrete problem given, derive relevant assessment criteria, survey existing implementations of these technologies, and assess these implementations using the assessment criteria derived. We also present results obtained in a prototype implementation using the winner solutions of our evaluation. This way, this paper represents a major step towards privacy-preserving attribute aggregation in federated eID systems.

This paper is structured as follows: Section 2 describes and analyses the problem addressed. Section 3 presents a survey of the building blocks proposed to solve the problem, while Sections 4 and 5 describe, respectively, the definition of the assessment criteria and the assessments performed. Section 6 shows the developed prototype and presents findings obtained from its implementation. Finally, Section 7 concludes this article and addresses remaining issues as well as possible future research directions.

[1] https://ec.europa.eu/digital-single-market/en/trust-services-and-eid

[2] https://www.eid-stork.eu/

[3] https://www.eid-stork2.eu/

2 Problem Analysis

The problem addressed in this paper is illustrated in Figure 1. Assume a relying party needs to access attributes from two attribute providers (e.g. AP_A and AP_B) from different identity systems (e.g. IS_A and IS_B). Assume further that the two attribute providers and the relying party all use different ontologies to represent attributes. Hence, the relying party cannot directly retrieve and use required attributes from the two attribute providers. To achieve attribute interoperability, the relying party relies on an Aligning Entity (AE) that acts as an intermediary between the relying party and the two attribute providers. The aligning entity can be regarded as a special kind of identity provider or attribute provider used by the relying party. It obtains attributes from different attribute providers, aligns and aggregates these attributes, and provides a consolidated set of attributes to the relying party.

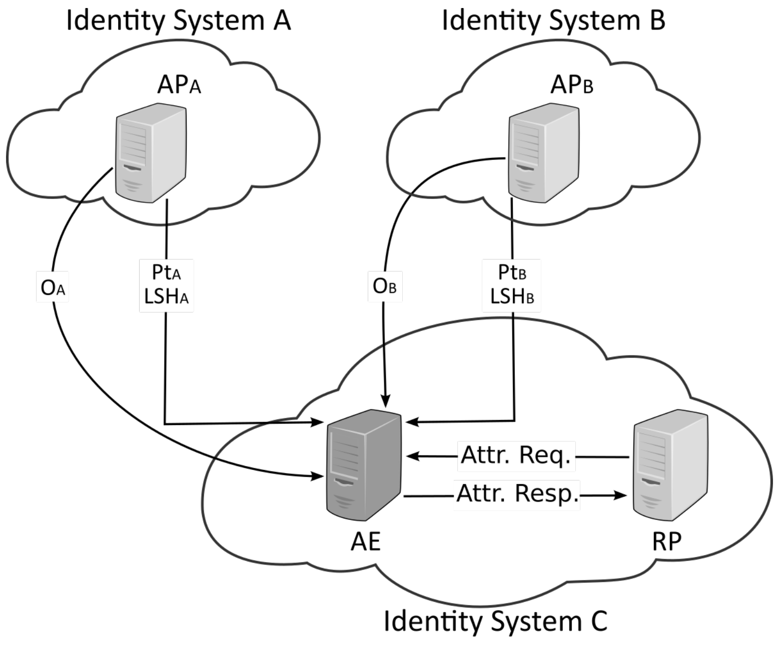


Figure 1. General architecture

To accomplish these tasks, the aligning entity needs to solve two problems. First, it must be able to handle different ontologies used by attribute providers from different identity systems (e.g. O_A and O_B). This ability is necessary, as the relying party and both attribute providers rely on different ontologies. Second, the aligning entity must not request more attributes from the two attribute providers in plain text (e.g. Pt_A and Pt_B) than originally requested by the relying party, to consider defined minimum-disclosure rules. Complying with the minimum-disclosure rules is a considerable challenge, as the aligning entity might require additional attributes to successfully handle different ontologies. If the aligning entity manages to solve these two fundamental problems, it can align and aggregate attributes represented by different ontologies while preserving privacy by maintaining defined minimum-disclosure rules concerning attributes.

In this paper, we propose the use of two different technologies to solve the problems defined above. We propose the use of ontology-alignment solutions and locality-sensitive hashing functions. In the following subsections we elaborate on these two technologies, motivate their use in the context of the problem defined, and discuss relevant related work.

2.1 Ontology-Alignment Solutions

Ontologies are a useful concept to facilitate the use of attributes from different attribute providers. Especially in closed eID systems, ontologies work well for this purpose, as they allow a higher degree of automation in the process of aligning and aggregating attributes from different attribute providers. However, a certain level of harmonization between different attribute providers is a mandatory requirement for the successful application of ontologies. In particular, all attribute providers need to use the same ontology, to assure that attributes represented by these ontologies can be aligned and aggregated. This degree of harmonization is usually not provided by attribute providers from different eID systems, as several identity systems usually use different ontologies to represent eID data and attributes. An intermediate entity is required to address this issue. This entity translates attribute ontologies into a form that can be understood by the relying party. In Figure 1, the aligning entity assumes the role of this entity.

In principle, the aligning entity can follow two approaches to achieve attribute interoperability between attribute providers and the relying party. First, following the ontology-merging approach, ontologies used by attribute providers are merged to a new ontology that is supported by the relying party. Nacer [1] discusses ontology-merging solutions in more detail. Second, following the ontology-alignment approach, the aligning entity creates an alignment describing the relation between different ontologies. In the example shown in Figure 1, the aligning entity would create an alignment that describes the mapping between the ontologies used by the two attribute providers and the one used by the relying party. When the relying party requests attributes from the attribute providers, the aligning entity uses the created alignment to transform attributes from the attribute providers into attributes that can be understood by the relying party.

In general, the ontology-alignment approach provides a higher degree of efficiency and flexibility. We hence focus on this approach for our solution. Section 3 surveys existing ontology-alignment solutions and implementations.

2.2 Locality-Sensitive Hash Functions

To be able to provide attributes to the relying party, the aligning entity potentially needs more attributes from the attribute providers than originally requested by the relying party. For instance, the aligning entity might need additional attributes to unambiguously identify a user. Unfortunately, requesting additional attributes from attribute providers potentially violates minimum-disclosure rules and hence reduces privacy.

To address this issue, we propose the use of locality-sensitive hash (LSH) functions. In contrast to ordinary hash functions, LSH functions reflect the similarity of input values to derived hash values. For instance, if inputs I_1 and I_2 have a similarity of degree S, then also hash values $H_1 = LSH(I_1)$ and $H_2 = LSH(I_2)$ have a similarity of degree S. The special characteristic of locality-sensitive hashing functions can be used to maintain the privacy of exchanged attributes. Concretely, attribute providers can provide the aligning entity with hash values of non-requested attributes instead of plain-text attributes. This way, the aligning entity is provided with the relevant information to provide the relying party with the attributes requested. Still, attributes not directly requested by the attribute provider remain unrevealed.

As an alternative to locality-sensitive hashing functions, zero-knowledge proofs [2] or homomorphic encryption [3] could be used as well. However, locality-sensitive hashing functions have already proven to be useful and applicable in various fields of application [4], [5]. Accordingly, there are several implementations of this concept. Hence, we focus on this technology to overcome privacy challenges. Section 3 provides a survey of existing implementations of locality-sensitive hashing functions.

3 Survey

The problem analysis conducted has revealed that attribute interoperability in eID federations requires two technologies, i.e. ontology-alignment solutions and locality-sensitive hashing functions. In this section we provide an overview of current implementations of these two technologies. The implementations surveyed will later be assessed using relevant criteria.

3.1 Existing Ontology-Alignment Solutions

An ontology constitutes an agreement for representing a common model to be shared among entities. This feature enables information exchange in a human-readable and understandable manner [6]. Ontology-merging and ontology-alignment approaches are used to obtain a common knowledge representation among entities (e.g. among different attribute definitions of attribute providers). Two or more ontologies are aligned to enable involved entities to use a common vocabulary to communicate with each other. The following paragraphs briefly sketch three of the most commonly used ontology-alignment solutions.

AlignAPI The Alignment API (AlignAPI[4]) is an API that can be used for representing alignments and for the development, integration, and composition of matchers. It is available in the Java programming language and provides examples and basic tools for manipulating alignments [7]. Additionally, the Alignment API provides a set of abstractions for expressing, accessing and sharing ontology alignments.

The Alignment API's reference implementation aims at facilitating the development of tools for manipulating alignments and calling matchers. It can also help matcher developers to deliver alignments in a well-supported framework [8].

XMAP The XMAP[5] ontology matching system is a high-precision system that can perform matching on large ontologies [9]. A semantic similarity measure is defined using UMLS[6] and WordNet[7] to provide a synonymy degree between two concepts in different ontologies, using their context.

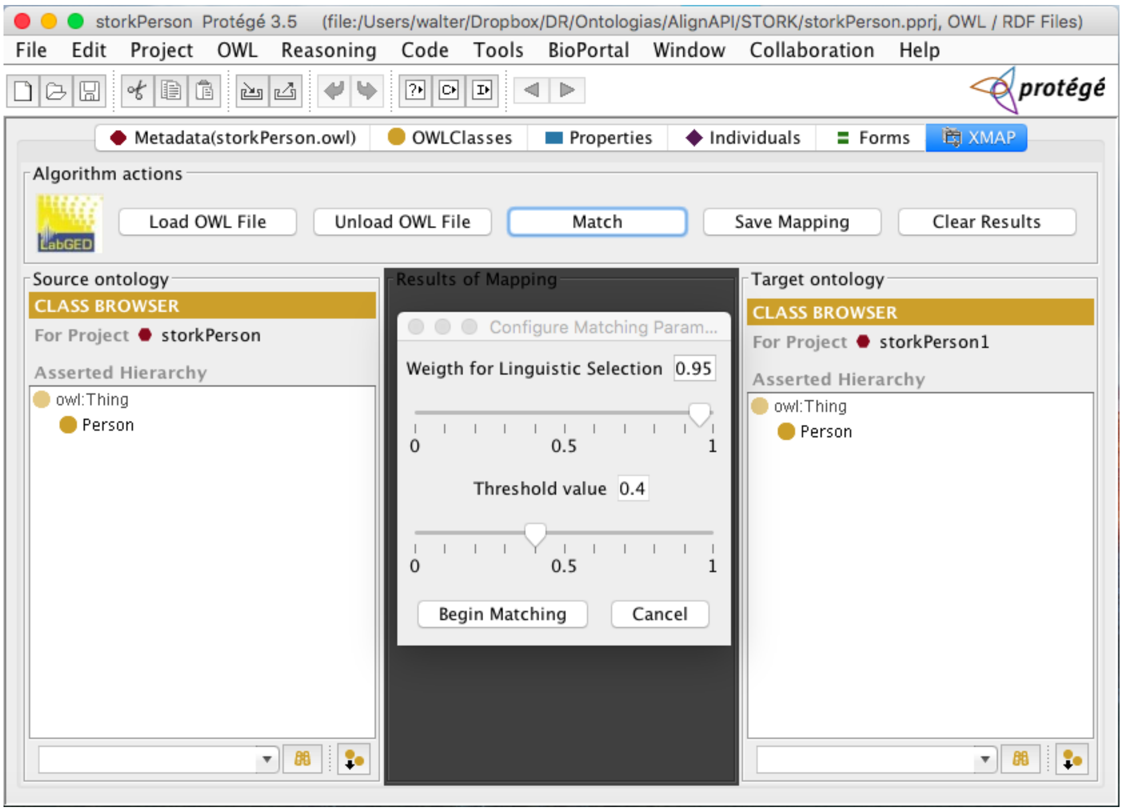
It relies on the Microsoft Translate API[8] to work with ontologies in many languages. It exploits the common elements (at linguistic and at structural levels) from the descriptions to measure the similarity between two classes and two properties respectively. Figure 2 shows the user interface of XMAP.

PROMPT PROMPT[9] is an algorithm and a tool for merging and aligning ontologies [10]. It requires direct interaction with the user. The tool takes two ontologies as input [11] and guides the user through the process of creating a merged/aligned ontology.

Initially, PROMPT creates a previous list of matches considering class names. Afterwards, it carries out the following steps in a loop:

1. The user interacts by either selecting one of PROMPT's suggestions or by editing the ontology to specify the desired operation directly.
2. PROMPT performs the operation, automatically makes the necessary changes based on the type of operation, generates a list of suggestions for the user based on the last operation, determines conflicts generated by the last operation, and finds solutions to those conflicts. Figure 3 shows the user interface of PROMPT.

[4] http://alignapi.gforge.inria.fr/
[5] http://www.labged.net/index.php?rubrique=mapage38
[6] https://www.nlm.nih.gov/research/umls/
[7] https://wordnet.princeton.edu/
[8] https://www.microsoft.com/en-us/translator/translatorapi.aspx
[9] http://protegewiki.stanford.edu/wiki/PROMPT



Weigth for Linguistic Selection 0.95

Threshold value 0.4

Figure 2. Main user interface of XMAP

3.2 Existing Locality-Sensitive Hashing Functions

Locality-sensitive hashing functions map similar objects into the same hash buckets with high probability. They perform a similarity query on an LSH index in two steps [12]:

1. Selecting candidate objects for a given query q using LSH functions; and
2. Ranking them according to their distances to q.

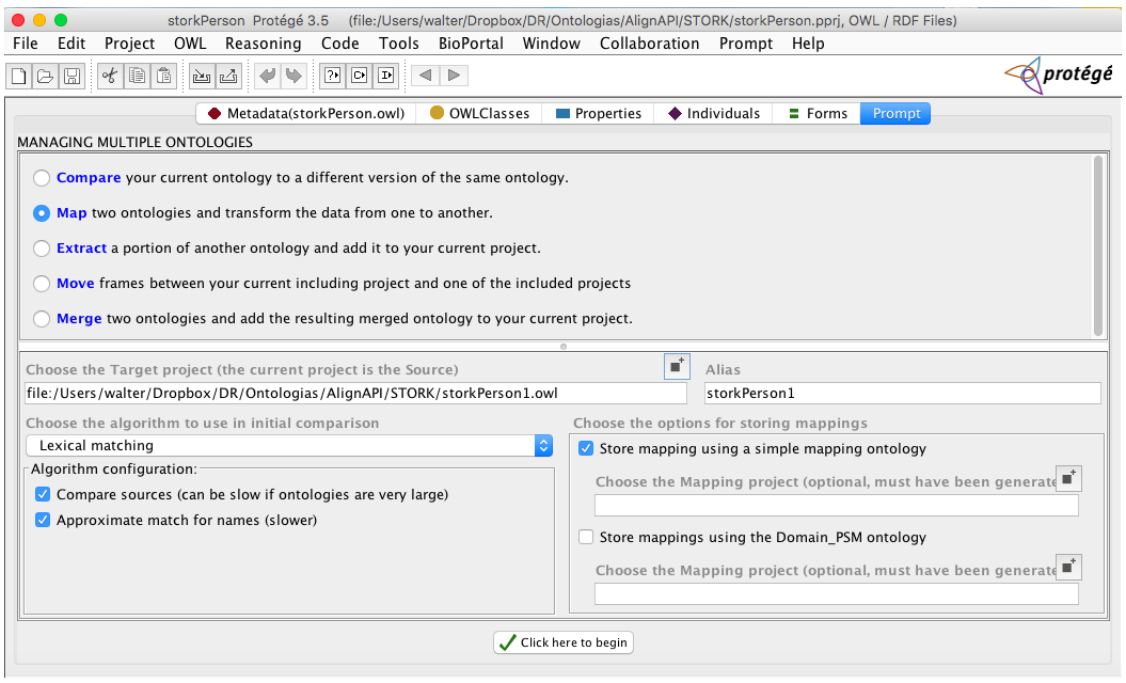
LSH functions ensure that the collision probability is higher for closer objects (objects with similar values) than for those that are far (objects with considerably different values) [5], [4]. In the following paragraphs, we briefly sketch existing implementations of LSH functions.

MinHash MinHash techniques evaluate the similarity of any two sets requiring only a constant number of comparisons [13]. MinHash works by extracting a representation $h_k(S)$ of a set S using a deterministic sampling. This $h_k(S)$ has a constant size k, independent from $|S|$. The computation of $h_k(S)$ incurs a complexity linear in set sizes.

Nilsimsa Nilsimsa [14] is a locality-sensitive hashing function that takes an arbitrary input and outputs an n-bit digest. It uses n buckets to count the trigrams that appear in the input and converts the counts to an n-bit digest. The similarity assessment between two inputs is performed comparing the corresponding positions of the two Nilsimsa digests and counting the number of equal bits. The algorithm counts the number of equal bits of the two Nilsimsa digests in the same position to identify the similarity between two inputs [15]. The higher is the number of equal bits, the more similar the two documents are.

Trend Locality Sensitive Hashing (TLSH) Summing up the distance[10] between the digest headers and the digest bodies determines the TLSH value. The resulting distance score ranges

[10] The distance is evaluated in a process similar to the hamming distance.


storkPerson Protégé 3.5 (file:/Users/walter/Dropbox/DR/Ontologias/AlignAPI/STORK/storkPerson.pprj, OWL / RDF Files)
File Edit Project OWL Reasoning Code Tools BioPortal Window Collaboration Prompt Help
protégé
Metadata(storkPerson.owl) OWLClasses Properties Individuals Forms Prompt

MANAGING MULTIPLE ONTOLOGIES

○ **Compare** your current ontology to a different version of the same ontology.

● **Map** two ontologies and transform the data from one to another.

○ **Extract** a portion of another ontology and add it to your current project.

○ **Move** frames between your current including project and one of the included projects

○ **Merge** two ontologies and add the resulting merged ontology to your current project.

Choose the Target project (the current project is the Source): file:/Users/walter/Dropbox/DR/Ontologias/AlignAPI/STORK/storkPerson1.owl

Alias: storkPerson1

Choose the algorithm to use in initial comparison: Lexical matching

Algorithm configuration:

✓ Compare sources (can be slow if ontologies are very large)

✓ Approximate match for names (slower)

Choose the options for storing mappings

✓ Store mapping using a simple mapping ontology

Choose the Mapping project (optional, must have been generate

☐ Store mappings using the Domain_PSM ontology

Choose the Mapping project (optional, must have been generate

✓ Click here to begin

Figure 3. Main user interface of PROMPT

from 0 to 1000+. Digests with a $distance \leq 100$ are considered to be similar. Digests with a $distance > 100$ shall be regarded as not being similar. The evaluation of the TLSH digest of the byte string follows these steps [16], [17]:

1. Process the byte string using a sliding window to populate an array of bucket counts.
2. Calculate the quartile points, q1, q2 and q3.
3. Construct the digest header values as a function of
   (a) the length of the file,
   (b) the quartile points calculated in step (2), and
   (c) a checksum.
4. Construct the digest body by processing the bucket array.
5. Generate the output digest by concatenating the digest header from step (3) and the digest body from step (4).

4 Assessment Criteria

The conducted survey on existing ontology-alignment solutions and locality-sensitive hashing functions has shown that there is already a variety of implementations available. Unfortunately, none of these implementations has been developed with the concrete use case of attribute interoperability in mind. It hence remains unclear, which of the implementations surveyed is best suited for realizing the aligning entity as depicted in Figure 1. To determine the best available implementations, we conduct systematic assessments on all solutions surveyed. As preliminary work, we define criteria in this section that are relevant to the use case given. Surveyed solutions will later be assessed using these criteria.

Most criteria are intentionally kept on a rather abstract level (e.g. Documentation) to achieve a common set of criteria applicable to all solutions surveyed. Still, all criteria have been defined such that they enable assessing surveyed solutions concerning their effectiveness (e.g. Processing Time, and Mappings Identified) and ease of integration (e.g. Implementation, License).

The effectiveness criteria are used to choose the winner solutions as they express quantitative metrics about the performance of all solutions.

The following subsections sketch all assessment criteria defined.

4.1 Common Assessment Criteria

Some of the criteria defined are relevant for both locality-sensitive hashing functions and ontology-alignment solutions. Most of them are rather non-technical, but still relevant especially with regard to a solution's ease of integration. In particular, the following criteria can be regarded as relevant for both ontology-alignment solutions and locality-sensitive hashing functions.

Documentation This criterion is related to the amount and quality of resources that are available to provide the knowledge required to apply the respective solution in a specific context. Examples for suitable resources are Wikipedia entries, project Wiki, web pages, GitHub repositories, or sample code. A comprehensive documentation is a consistent basis for a successful integration of provided implementations into own solutions.

Implementation Implementation details, e.g. the programming language used or interfaces provided, is another relevant criterion that needs to be taken into account, as it influences the complexity of integrating provided implementations into own solutions. For instance, implementations relying on popular and widely used programming languages are probably easier to integrate into own solutions. Furthermore, availability of an API or of well-defined plug-ins can also increase ease of integration.

License The license of the respective implementation is another relevant criterion, as it defines limitations in reusing provided functionality for own solutions. Hence, this criterion can also be regarded as a factor that influences ease of integration.

Processing Time The processing time describes how much time is required to execute a given task according to the measured smallest value unit such as ms, s. The processing time can be a relevant criterion if performance and efficiency are important aspects.

4.2 Assessment Criteria for Locality-sensitive Hashing Functions

In addition to the criteria that are relevant for both ontology-alignment solutions and locality-sensitive hashing functions, several criteria can be identified that are unique for locality-sensitive hashing functions. The following subsections depict these criteria.

Function Score and Clear-Text Score The Function Score (F-S) and the Clear-Text Score (CT-S) provide the average and the total, respectively, similarity score in a given test. The values are computed using the Levenshtein Distance between the values provided as input and are normalized considering the signature length.

Signature Length The Signature Length represents the output length of the respective locality-sensitive hashing function. Similar to ordinary hash functions, the output length is usually fixed. If the solution that aims to integrate locality-sensitive hashing functionality has specific restrictions, the output length can be a relevant criterion.

Similarity Scale This criterion is related to the result values yielded by the particular solution to give a metric on the similarity of the evaluated inputs. Usually, each solution uses a different scale. Common examples are, e.g. [0.0–1.0], [0–10], [128–(-128)], etc. The similarity scale used and supported can be a relevant criterion when integrating the respective implementation into an own solution.

Restrictions This criterion is defined to provide some specific conditions on using the evaluated implementation. The values are textual descriptions of these particularities, which might be relevant for solutions that aim to rely on restricted implementations.

4.3 Assessment Criteria for Ontology-Alignment Solutions

Finally, specific assessment criteria can be defined for ontology-alignment solutions as well. The following paragraphs describe them in more detail.

Similarity Scores The Similarity Scores provide the values, considering the Similarity Scale of the evaluated solution, obtained from the assessment, e.g. 1.0, 0.85. They provide feedback on how much certainty the solution has on an identified mapping.

Mappings Identified The Mappings Identified metric provides the absolute number of correspondences on both evaluated ontologies. It is useful to identify the total amount of concepts that have a correspondence on evaluated ontologies.

5 Assessment Results

In this section the assessment criteria defined in Section 4 are mapped to the solutions surveyed in Section 3. This way, those solutions are identified that satisfy best relevant requirements and are hence best suited to be used for realizing Aligning Entities in eID federations as depicted in Figure 1. The results' evaluation considers the best ontology alignment solution, which promotes the interoperability. Moreover, it also considers the best locality-sensitive hashing function, which improves the ontology alignment step. The following subsections present the assessments results obtained.

5.1 Assessment Results of Ontology-Alignment Solutions

All surveyed ontology-alignment solutions come with diverse sources of documentation (e.g. OD – online documentation, T – tutorial, WP – web page, W – WikiPedia). Furthermore, all surveyed implementations are available either as Java APIs or as Protégé Plugins. They have been published either under GNU Lesser General Public License (LGPL) or Mozilla Public License (MPL). All results concerning available documentation and implementation are summarized in Table 1. It is important to notice that the XMAP solution does not provide any information about licensing on its web page.

Table 1. Documentation, Implementation, and License assessments

Solution	Documentation	Implementation	License
AlignAPI	OD, T	Java API	LGPL
PROMPT	W	Java API, Protégé Plugin	MPL
XMAP	WP (last update 2011)	Protégé Plugin	N/A

In addition to comparing the ontology-alignment solutions surveyed by means of their meta information as depicted in Table 1, we also ran some tests to compare their effectiveness and efficiency. The tests loaded the two eID-related ontologies, O_1 and O_2, and verified the obtained matches. These tests considered each concept and class, according to the particular solution's implemented approach. The two ontologies used have had the same concepts, but four of them had been written with similar terms, namely: eMail $\leftrightarrow$ e-mail; dateOfBirth $\leftrightarrow$ birthDay; surname $\leftrightarrow$ lastName; and givenName $\leftrightarrow$ name. Since the ontologies O_1 and O_2 present previously known similar terms, we were able to verify the accuracy of each ontology-alignment solution. Results obtained are summarized in Table 2. At this point, it is important to note that the performance of the solution XMAP could not be assessed in detail, as no successful alignment process could be completed.

In addition to measuring their accuracy, we also used the tests executed on the surveyed solutions to investigate their processing time. Running the test with the ontologies O_1 and O_2 (both available online as storkPerson[11] and storkPerson1[12]) has shown that the solution AlignAPI is 2.75 times faster than ontology-alignment solution PROMPT.

[11] http://web.tecnico.ulisboa.pt/walter.filho/ontologies/STORK/storkPerson.owl
[12] http://web.tecnico.ulisboa.pt/walter.filho/ontologies/STORK/storkPerson1.owl

Significant differences between the two solutions AlignAPI and PROMPT have also been obtained regarding the number of matchings found. Running tests with the two ontologies O_1 and O_2 yielded 22 matchings found by the AlignAPI and only one match found by PROMPT. These results are also reflected by Table 2.

Summarizing the results obtained from assessing the surveyed ontology-alignment solutions, it follows that AlignAPI is the most suitable solution. This solution outperforms other alternatives regarding performance and ease of integration.

Table 2. Processing time, Similarity Scores, and Matchings assessments

Solution	Processing Time	Similarity Scores	Matchings Identified
AlignAPI	1.467s	0.96807	22
PROMPT	4.036s	Not provided	1
XMAP	Not finished	N/A	N/A

5.2 Assessment Results of Locality-Sensitive Hashing Functions

Similar to the ontology-alignment solutions, all surveyed locality-sensitive hashing functions provide diverse sources of documentation (e.g. G – GitHub, S – Sample Code, W – WikiPedia). They are available in several programming languages like Java, Python, Ruby, PHP, Go, C (Group 1); and C++, C# (Group 2). All surveyed solutions are available under Apache License (AL). The Similarity Scale[13] of each solution was evaluated as well. Obtained results are summarized in Table 3.

Table 3. Documentation, Implementation, License, and Similarity Scale assessments

Solution	Documentation	Implementation	License	Sim. Scale
MinHash	GSCW	Group 1	AL	0 - 10
Nilsimsa	GSCW	Group 1 + Group 2	AL	128 - (-128)
TLSH	GitHub	Python, Java, JS	AL	0 - 200 / 0 - 400

Again, we did not limit our assessments and comparisons to the criteria covered by Table 3. We also ran tests to assess the performance of the different solutions. Note that the TLSH[14] function is unable to produce results on inputs with a size smaller than 256 bytes. This feature makes this solution unsuitable for use cases related to eID attributes, as eID attributes are potentially rather short. For this reason, TLSH was precluded from further performance assessments.

Four tests have been defined and run to assess the performance of the remaining two locality-sensitive hashing solutions. All four tests used the two locality-sensitive hashing solutions to compute hash values from user names, i.e. a typical eID attributes. A set of 1,000 records with random data (such as given name, family name, birthday) was generated and used to run these tests. For each test, slightly different input values have been used as defined below:

Test 1: The same full user name was provided to both of the two locality-sensitive hashing solutions.

Test 2: Solution A was provided with the full user name. Solution B was provided with the same full user name but using some abbreviation within the name.

[13] The similarity scale of TLSH function can vary depending on the signature's size. For signatures with size 70, it goes from 0–200. Moreover, for signatures with size 134, it goes from 0–400.

[14] https://github.com/trendmicro/tlsh

Test 3: A set of 1,000 same full user names was provided to both of the two locality-sensitive hashing solutions.

Test 4: Solution A was provided with a set of 1,000 full user names. Solution B was provided with the same full user names but using some abbreviations within the names.

Execution of Test 1 allowed obtaining the metrics of each function performing the analysis of just one element with a full match. It represents the scenario where a user authenticates providing its data and the attribute provider compares data provided with unique user data it has already stored. Test 2 provides a similar metric but considers similar values, not the exact values on both sides. Test 3 provides a similar result of Test 1 but uses a set of 1,000 records on the attribute provider. This way, Test 3 demonstrates how the assessed functions perform in a situation that is close to a real-world scenario. The same applies to Test 4, which combines the specifics of Test 2 and Test 3. Table 4 shows the results obtained from running theses tests.

Table 4. Test results for LSH Functions MinHash and Nilsimsa

Solution	**Test 1**			**Test 2**			**Test 3**			**Test 4**		
	Time	**F-S**	**C-S**	**Time**	**F-S**	**C-S**	**Time**	**F-S**	**C-S**	**Time**	**F-S**	**C-S**
MinHash	90.730	0.941	0	94.080	0.941	5	597.4	0.847	0	518.7	0.847	0.288
Nilsimsa	2.074	0.000	0	3.169	0.560	5	278.5	0.000	0	327.0	0.667	0.288

The processing time of the locality-sensitive hashing function Nilsimsa was from 43x (Test1) to 1.6x (Test 4) faster than the processing time of the MinHash function. Beside processing time, the relation between the similarity of input values and output values is another relevant quality measure for LSH functions. We used the Levenshtein Distance (LD) to estimate the similarity of input values and output values. Results obtained show that the Nilsimsa LSH function yielded closer values to the LD of input text than the MinHash LSH function. For Test 1 and Test 3, the Nilsimsa's Function Score (F-S) was the same in both tests. For Test 4, the result of the Nilsimsa LSH function was 25% closer than the results of the MinHash function. This is also reflected by Table 4.

From the assessment results of the surveyed locality-sensitive hashing Functions shown in Table 4, it can be concluded that Nilsimsa is the clear winner. This LSH solution outperforms other surveyed and assessed solutions and complies best with relevant assessment criteria identified.

6 Prototype

To evaluate the feasibility of the identified winner solutions, we have developed a proof-of-concept prototype implementation.

6.1 Implementation

The implementation realizes the two entities, relying party and attribute provide,r as illustrated in Figure 4 using RESTful Web Services (WS) written in Java using the JAX-RS RESTful API[15]. Our implementation focuses on the relying party and attribute provider intentionally. Implementation of the intermediate gateways is regarded trivial. Respective solutions are already available, e.g. STORK [18].

In the presented implementation, the AlignAPI performs the Ontology Alignment (First Step), and the Nilsimsa locality-sensitive hashing function blinds attribute values (Second Step) used to increase the level of assurance on the obtained alignment. Additionally, the prototype uses the

[15] https://jersey.java.net/index.html

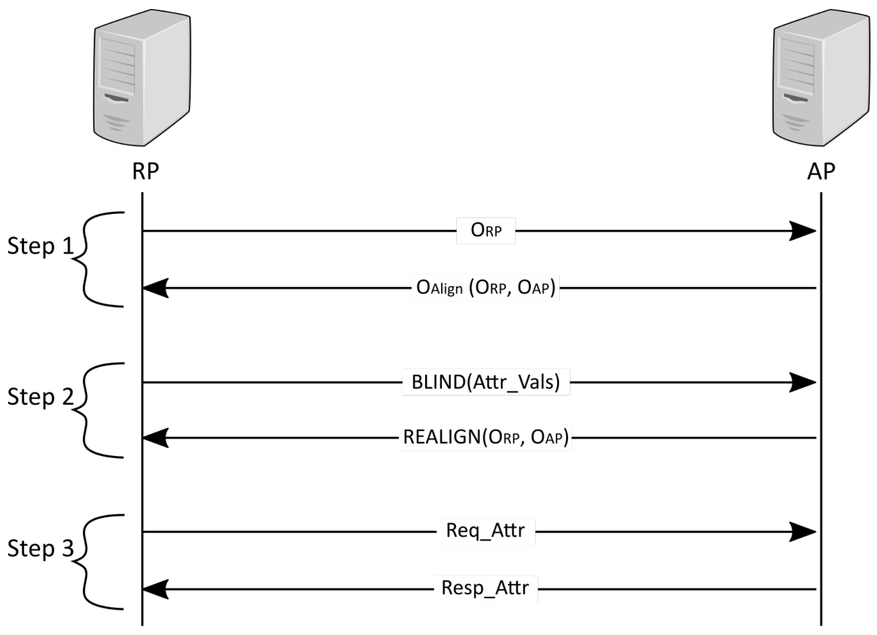


Figure 4. Communication overview

ontop[16] platform because it provides an interface between the ontologies used and the data stored in the database, enabling to query ontologies using SPARQL.

Step 1: Ontology Alignment In the Ontology Alignment step, the relying party submits its ontology (O_{RP}) to the attribute provider. The attribute provider uses AlignAPI to align O_{RP} with its own ontology yielding $O_{ALIGN} = O_{AP} \cap O_{RP}$. This alignment produces a Resource Description Framework (RDF) file. The RDF file contains all attribute-name pairs and their corresponding confidence levels (CLs) identified by the AlignAPI. These CLs are taken observing a threshold provided during the ontology-alignment process.

Step 2: Alignment Improvement In this step, the attribute provider tries to improve the confidence levels of the attribute-name pairs from Step 1. Note that performing this step requires the relying party to know some attribute values. Knowing theses attribute values is a valid assumption, as the relying party can obtain required values, e.g. from the user or another local attribute provider.

The attribute provider requests from the relying party blinded attribute values for all attributes in O_{ALIGN} with confidence levels (CLs) smaller than 100%. The relying party receives an eID value and the attribute name as input through a web service public interface and returns the corresponding blinded attribute value. The attribute provider executes a similar process, blinding its own stored attribute value, to assess the similarity with the obtained blinded value retrieved from the relying party.

The similarity (sim) of the blinded attribute values is assessed using the Nilsimsa Distance (ND). Then, sim is normalized on a 0-1 scale. If sim is greater than a given threshold (e.g. $sim > 0.98$), it improves the confidence level (CL) to 100%. Otherwise, for sim values between 0.97 and 0.64 an increment factor (IF) is defined proportionally,

$$CL = CL + (sim * IF) \tag{1}$$

e.g. if sim is between 0.65 and 0.84, the CL value is updated to $CL = CL + (sim * 0.125)$.

After updating the respective confidence level (CL) values for each attribute-name pair, the attribute provider returns the improved alignment to the relying party. The relying party concludes the alignment-improvement step.

[16] http://ontop.inf.unibz.it/

Step 3: Attribute Exchange The conducted ontology-alignment process enables relying party and attribute provider to exchange attributes, while both entities can rely on their own terminology. With the help of the implemented WS interface, the relying party can request attributes based on its own ontology, i.e. O_{RP}. The produced ontology alignment is used to map the attributes required by the relying party, using O_{RP}, to the nomenclature utilized by the attribute provider, i.e. O_{AP}. This way, the attribute provider can perform a query on its database using its terminology. The attribute provider uses the parameters of the relying party's request to parametrize an SPARQL query. This query is then translated, using the ontop platform, into an SQL query and executed on the attribute provider's local database. Finally, the attribute provider sends back to the relying party the resulting set of attribute names and values (Figure 4, Step 3).

6.2 Results

To evaluate our prototype, we experimented with different ontologies each containing ten attributes and their respective attribute values. This way, we investigated the capabilities of our prototype to deal with different input data. In total, we used 10 ontologies (O_1–O_{10}) with similar terminology, as described in Section 5.

For our experiments, the attribute provider uses O_1 to represent stored eID attributes. The remaining nine ontologies (O_2–O_{10}) were assigned to the relying party. This way, we were able to conduct nine test runs evaluating the proposed solution's capability to align different ontologies. The attribute values used in the assessment where generated by a random data generator[17].

In our experiments we used two metrics to assess our prototype's performance: precision and confidence level (CL). The precision [19] is obtained from:

$$P = \frac{T_P}{T_P + F_P} \tag{2}$$

and the CL is the probability that two attribute names of the respective attribute-name pair correspond to each other. The solution under evaluation provides the CL value used in our assessment. The precision and CL values presented in Table 5 are the average values of the whole set of attribute-names (Step 1) and the average values of attribute-values (Step 2) to each ontology (O_2 to O_{10}) assessed with the respective sets in the Ontology 1 (O_1).

The results obtained show that our prototype achieves confidence values of 100% for all tested ontologies in the second alignment step. The results obtained and illustrated in Table 5 show that

Table 5. Obtained results from the experiments

	Step 1		Step 2	
	PAp (%)	CA (%)	PNp (%)	CN (%)
O_2	100.00	47.00	100.00	100.00
O_3	90.91	44.00	100.00	100.00
O_4	47.62	9.00	83.33	100.00
O_5	100.00	47.00	100.00	100.00
O_6	83.33	33.00	100.00	100.00
O_7	83.33	33.00	100.00	100.00
O_8	21.28	9.00	90.91	100.00
O_9	83.33	33.00	100.00	100.00
O_{10}	19.95	9.00	83.33	100.00

[17] https://mockaroo.com/

the second alignment step increased the precision from 21.28% up to 90.91% for Ontology 8. For Ontology 2 and Ontology 5, precision was already at 100% after the first step. The second alignment step increased the confidence levels for all ontologies up to 100%.

7 Conclusions and Future Work

In this paper, we have proposed the use of ontology alignment and locality-sensitive hashing functions to leverage attribute interoperability in eID federations. A survey conducted has revealed that implementations of these two technologies already exist but are not tailored to use cases related to eID federations. We have hence applied systematic assessments of available solutions using a set of requirements. These assessments have revealed that AlignAPI [7] is the most suitable ontology-alignment solution available. Furthermore, Nilsimsa [14], [15] turned out to be the locality-sensitive hashing implementation that meets best the special requirements of eID federations.
A prototype was implemented to verify the feasibility of our Alignment Entity proposal. Through its implementation, we conducted experiments using ten different ontologies and ten data sources with the same attribute values, but different terminologies. Obtained results show that it is possible to establish alignments between distinct attribute names using the winner solutions identified by our assessments. We also found out that the second step proposed increases the confidence level of the alignment in the same ratio as the attribute-values are similar.
In future work, we will use the assessment results obtained to enhance the prototype implementation of the aligning entity. This enhanced aligning entity will finally enable attribute interoperability in eID federations such as the European STORK framework. We are also going to implement a history feature on the assessment of the similarity between ontologies. This way, it will be possible to increase/decrease the confidence on the obtained alignments considering also their past performance.
The work presented in this paper is a fundamental basis for the realization of the aligning entity, as it assures that its two most important building blocks, i.e. ontology alignment and locality-sensitive hashing functionality, are implemented in the most effective and efficient way.

Acknowledgment

This work was partially supported by CAPES Proc. Num. BEX 9096/13-2 and EU project Stork 2.0 CIP-ICT-PSP-2011-5-297263.

References

[1] H. Nacer and D. Aissani, “Semantic Web Services: Standards, Applications, Challenges and Solutions,” Journal of Network and Computer Applications, vol. 44, pp. 134–151, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2014.04.015

[2] F. Paci, R. Ferrini, A. Musci, K. Steuer, and E. Bertino, “An Interoperable Approach to Multifactor Identity Verification,” Computer, vol. 42, no. 5, pp. 50–57, May 2009. [Online]. Available: http://dx.doi.org/10.1109/MC.2009.142

[3] P. Boufounos and S. Rane, “Secure Binary Embeddings for Privacy Preserving Nearest Neighbors,” in Information Forensics and Security (WIFS), 2011 IEEE International Workshop on, Nov. 2011, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/WIFS.2011.6123149

[4] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, “Locality-Sensitive Hashing Scheme Based on P-Stable Distributions,” in Proceedings of the Twentieth Annual Symposium on Computational Geometry, ser. SCG ’04. New York, NY, USA: ACM, 2004, pp. 253–262. [Online]. Available: https://doi.org/10.1145/997817.997857

[5] P. Indyk and R. Motwani, “Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality,” in Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, ser. STOC ’98. New York, NY, USA: ACM, 1998, pp. 604–613. [Online]. Available: https://doi.org/10.1145/276698.276876

[6] L. Hind, D. Chiadmi, and L. Benhlima, “How Semantic Technologies Transform E-Government Domain,” Transforming Government: People, Process and Policy, vol. 8, no. 1, pp. 49–75, 2014. [Online]. Available: http://dx.doi.org/10.1108/TG-07-2013-0023

[7] J. David, J. Euzenat, F. Scharffe, and C. T. dos Santos, “The Alignment API 4.0,” Semantic Web journal, vol. 2, pp. 3–10, 2011.

[8] J. Euzenat, “An API for Ontology Alignment,” pp. 698–712, 2004. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30475-3_48

[9] W. E. Djeddi, M. T. Khadir, and S. B. Yahia, “XMap: Results for OAEI 2015.” [Online]. Available: http://ceur-ws.org/Vol-1545/oaei15_paper17.pdf

[10] S. Malik, N. Prakash, and S. Rizvi, “Ontology Merging Using Prompt Plug-In of Protégé; in Semantic Web,” in Computational Intelligence and Communication Networks (CICN), 2010 International Conference on, Nov. 2010, pp. 476–481. [Online]. Available: http://dx.doi.org/10.1109/CICN.2010.151

[11] N. F. Noy and M. A. Musen, “Algorithm and Tool for Automated Ontology Merging and Alignment,” in Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831, 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=647288.721118

[12] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, “Multi-probe LSH: Efficient Indexing for High-dimensional Similarity Search,” in Proceedings of the 33rd International Conference on Very Large Data Bases, ser. VLDB ’07. VLDB Endowment, 2007, pp. 950–961. [Online]. Available: http://dl.acm.org/citation.cfm?id=1325851.1325958

[13] C. Blundo, E. De Cristofaro, and P. Gasti, “EsPRESSo: Efficient Privacy-Preserving Evaluation of Sample Set Similarity,” in Data Privacy Management and Autonomous Spontaneous Security, ser. Lecture Notes in Computer Science, R. Di Pietro, J. Herranz, E. Damiani, and R. State, Eds. Springer Berlin Heidelberg, 2013, vol. 7731, pp. 89–103. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35890-6_7

[14] J. Zhang, H. Lu, X. Lan, and D. Dong, “DHTnil: An Approach to Publish and Lookup Nilsimsa Digests in DHT,” in High Performance Computing and Communications, 2008. HPCC ’08. 10th IEEE International Conference on, Sept 2008, pp. 213–218. [Online]. Available: http://dx.doi.org/10.1109/HPCC.2008.26

[15] Z. Jianzhong, Y. Boyang, L. Hongbo, and L. Xiaofeng, “PeerNil: An Approach to Publish and Lookup Nilsimsa Digest in Chord,” in Communications and Networking in China, 2008. ChinaCom 2008. Third International Conference on, Aug 2008, pp. 202–207. [Online]. Available: http://dx.doi.org/10.1109/CHINACOM.2008.4685003

[16] J. Oliver, C. Cheng, and Y. Chen, “TLSH – A Locality Sensitive Hash,” pp. 7–13, Nov. 2013. [Online]. Available: http://dx.doi.org/10.1109/CTC.2013.9

[17] A. Azab, R. Layton, M. Alazab, and J. Oliver, “Mining Malware to Detect Variants,” in Cybercrime and Trustworthy Computing Conference (CTC), 2014 Fifth, Nov. 2014, pp. 44–53. [Online]. Available: http://dx.doi.org/10.1109/CTC.2014.11

[18] V. Koulolias, A. Kountzeris, H. Leitold, B. Zwattendorfer, A. Crespo, and M. Stern, “STORK E-Privacy and Security,” Proceedings - 2011 5th International Conference on

Network and System Security, NSS 2011, pp. 234–238, 2011. [Online]. Available: http://dx.doi.org/10.1109/ICNSS.2011.6060006

[19] C. Goutte and E. Gaussier, A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 345–359. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-31865-1_25