MERA: A Musical Entities Reconciliation Architecture Based on Semantic Technologies

Daniel Fernández-Álvarez, Department of Computer Science, University of Oviedo, Oviedo, Spain

Jose Emilio Labra Gayo, University of Oviedo, Oviedo, Spain

Daniel Gayo-Avello, Department of Computer Science, University of Oviedo, Oviedo, Spain

Patricia Ordóñez de Pablos, Department of Business Administration. Faculty of Economics and Business, University of Oviedo, Oviedo, Spain

ABSTRACT

In this paper, the authors describe Musical Entities Reconciliation Architecture (MERA), an architecture designed to link music-related databases adapting the reconciliation techniques to each particular case. MERA includes mechanisms to manage third party sources to improve the results and it makes use of semantic technologies, storing and organizing the information in RDF graphs. They have implemented a prototype of their approach and have used it to link sources with different levels of data quality. The prototype has been effective in more than 94% of the cases under the conditions of their experiments. The authors have also compared their prototype with a well-known music-specialized search engine, outperforming the search results in the two experiments that they performed.

KEYWORDS

Adaptable Architecture, Collective Matching, Entity Reconciliation, Music Metadata, RDF Graph, Record Linkage, Semantic Search

1. INTRODUCTION

Although the problem of entity reconciliation has been largely studied, it remains a challenging issue. The proliferation of large databases with potentially repeated entities across the World Wide Web drives into an interest to find methods to detect duplicated entries when no reliable unique identifiers are available. In this paper, we provide an architecture for the specific task of linking records of musical databases. The Musical Entities Reconciliation Architecture (MERA) discovers links between elements of different databases that represent the same real-world entity in the music domain. Our approach is able to adapt the linkage process to the different content and nature of each database, letting the user configure different reconciliation algorithms for different attributes or type of entities.

Examples of fields usually contained in musical databases are titles, artist names, albums, genres, etc. The task of recognizing these kinds of contents is strongly connected to the record linkage problem, since it consists of the detection of records or entries referring to the same real-world entity. However, we have designed MERA with the assumption that the type of metadata linked to the music world presents a certain number of peculiarities that should be considered. For instance, there are many

DOI: 10.4018/IJSWIS.2017100103

specific cases of correct forms, or at least recognizable forms, in which we could express the name of an artist, including but not limited to:

- Artistic names vs civil names, e.g., "Stefani Joanne Angelina Germanotta" or "Lady Gaga";
- Naming conventions, e.g., "The Beatles" or "Beatles, the";
- Official or widely extended alias, e.g., "The King of Rock" instead of "Elvis Presley";
- Mixings between civil names and artistic names, e.g., "Shakira", "Shakira Isabel Mebarack Ripoll", "Shakira Mebarack", etc.;
- Acronyms, e.g., "System of a down" or "SOAD";
- Usual misspellings, e.g., "Bruce Springsting" instead of "Bruce Springsteen";
- Name of an artist linked to a song that should actually be linked to a group, e.g., "Michael Jackson" instead of "The Jackson Five".

Issues such as misspellings or acronyms are not specific of music-related metadata, and they can be found in databases or datasets of different nature. However, issues such as the existence of both artistic and civil name are exclusive of artists' specification. By contrast, when trying to conciliate other types of musical entities, e.g., songs, a different set of specific problems related to the nature of songs may appear. An example could be the management of the word "feat" (or variations such as "ft.", "featuring", etc.). When "feat" appears in a song title, it usually means that in that title there is a name of a collaborator included. Both "feat" and its following words may be discarded from the song name itself. However, they can possibly be computed in some other way since they can become very useful information.

In noisy or hand-made databases it is also possible to find extra words at the beginning or at the end of a song title. For instance, sequences linked to the radio program in which a song was played or to the place of a live performance. This may become even more troublesome in especially noisy databases such as those formed by the compilation of standalone audio files' metadata. When handling audio files wrongly labeled, it is possible to find titles that in fact contain all the associated metadata (artist, date, genre...) in a single field.

Another example of a musical concept that presents associated issues due to its special nature is the genre. When dealing with genres, it could happen that the same song is specified as pop in a database, as rock in a second one and as pop-rock in a third one. Sometimes, the same genre is even named with different forms that are in fact expressing the same reality.

Our assumption is that finding general reconciliation rules between two databases is far from being trivial, as well as finding appropriate rules or strategies to conciliate each field of those databases. The result could drastically change if it is compared to the rules that may be used when handling a different pair of sources. Trying to establish general rules could drive into an unnecessary number of failures (false positives/negatives) when identifying two records of different databases as forms of the same real entity. The inference of reconciliation rules in a particular case through the use of training data may be handy for covering issues such as misspellings, naming conventions or even noisy prefixes/suffixes, but they cannot handle cases in which the strings that represent the entities do not have common characters (example: "The King of Rock" should be recognized as "Elvis Presley").

Our main contribution in this paper is the specification of MERA architecture. MERA tries to adapt to all those scenarios using graph concepts and semantic web technologies. Our approach turns the information of one of the target databases into a custom RDF graph G containing all the information (name variations, alias, common misspellings, etc.) of every database record, as well as the relations between those records. The records of the second database are turned into complex queries that will be launched against *G*. The result of each query is the list of the most similar nodes to the target record according to:

- String distance based functions;
- Use of all the alternative identifying forms of a concept;
- Graph navigation in order to detect shared associated entities for disambiguation purposes.

MERA can use different reconciliation algorithms for each pair of databases and even for each field of those databases, trying to cover all the issues linked to the nature of the data. It is able to reach better results with more prior knowledge of the data issues, since the user is the agent that specifies the algorithms to use. MERA allows configuration about different properties that should be considered, the reconciliation algorithms to apply in each case and the threshold of similarity that a result must reach to be accepted. It also provides mechanisms to incorporate ad-hoc algorithms in the reconciliation process.

Our approach is designed to involve several sources at a time during the linking process by merging them in a single and reusable RDF graph. MERA describes a graph schema in which every piece of information is qualified with its original source through reification techniques. This allows the algorithm to distinguish which nodes of the graph are potential matches between a given pair of sources and which ones are merely used to enrich the data and improve the matching process. This also allows ignoring the content of certain sources in another use of the built graph.

In section 2, we make a revision of the state of the art of record linkage and we describe some related works. In section 3, we provide a detailed overview of MERA. In section 4, we describe, perform, and discuss several experiments using a prototype of MERA and we compare its effectiveness with a baseline approach. Finally, in section 5, we enumerate the conclusions of our work as well as future research lines. The source code is available at https://github.com/DaniFdezAlvarez/wMERA.

2. STATE OF THE ART

Record linkage, also referred as object identification (Tejada, Knoblock, & Minton, 2001, 2002), datacleaning (Do & Rahm, 2002), approximate matching or approximate join (Gravano et al., 2001; Guha, Koudas, Marathe, & Srivastava, 2004), fuzzy matching (Ananthakrishna, Chaudhuri, & Ganti, 2002), entity resolution (Benjelloun et al., 2009), reference reconciliation (Dong, Halevy, & Madhavan, 2005), or coreference resolution (Lee et al., 2013; Ng & Cardie, 2002), is a widely-studied problem. Nevertheless, with the proliferation of huge databases in the Era of Big Data and the need of developing effective and scalable reconciliation systems, the scientific community still put much effort to solve the challenges of record linkage (Enríquez, Domínguez-Mayo, Escalona, Ross, & Staples, 2017). The essence of the problem consists of identifying two or more elements that refer to the same reality. Basic use cases are the detection of duplicate entries within a file or the detection of equivalents across two databases.

Research work about entity resolution has been largely based on Fellegi & Sunter (1969), which was inspired by the ideas introduced by Newcombe & Kennedy (1962). Record linkage is presented as a classification problem, where a pair of entities can be classified as “matching” or “non-matching”. Several scientific communities have adopted that scheme to formulate the problem in its own way, producing many reusable techniques and technologies to solve it (P Christen, 2012; Winkler, 2014). A systematic study of the type of techniques developed in the last seven years for reconciliation tasks in Big Data environments is provided in (Enríquez et al., 2017).

Since record linkage becomes a problem due to the lack of unique reliable identifiers, traditional approaches are highly based in string comparators. Being able to recognize different strings that represent the same real-world object has been, and still is, a major research project (Hall & Dowling, 1980; Navarro, 2001; Yu, Li, Deng, & Feng, 2016). Despite there have been many works defining string similarity measures (Chaudhuri, Ganjam, Ganti, & Motwani, 2003; Cohen, Ravikumar, & Fienberg, 2003; Monge, Elkan, & others, 1996; Navarro, 2001), it has been concluded that there is not such an algorithm or combination that can outperform all the rest in terms of accuracy and efficiency in every context (Cohen et al., 2003; Harron, Goldstein, & Dibben, 2015), not even if we try to compare specific subsets of algorithms such as string-edit distance metrics(Peng, Li, & Kennedy, 2014).

Selecting the most accurate strategy to apply in order to get the best possible results is not a trivial task. Some research lines have put efforts in the design of methods to automatically detect which algorithm or combination of algorithms from a known set of possibilities works better for a particular scenario (Bilenko & Mooney, 2003; Nguyen & Ichise, 2016; Sarawagi & Bhamidipaty, 2002). This can be done through providing training data containing a set of pairs qualified as "matching" or "non-matching" and applying automatic learning techniques.

The techniques used to determine the similarity between two records through applying string distance metrics over their attributes are known as content-based or Feature-Based Similarities (FSB). However, sometimes FBS is not enough to properly if two entities match the same real object, especially when a disambiguation is needed (Kalashnikov & Mehrotra, 2006). In those scenarios, checking relations between entities in addition to entities' features may be a mechanism to improve matching. Traditional FBS approaches match each individual independently. By contrast, approaches in which relations between records are considered to produce a result are named context-based (Rahmani, Ranjbar-Sahraei, Weiss, & Tuyls, 2016) or collective entity resolution systems (Bhattacharya & Getoor, 2007). The representation of relations between entities fits well in graph structures, so these kinds of approaches are usually graph-based (P Christen, 2012). Those graph-based approaches combined with FBS can outperform the matching quality of FBS standalone on different scenarios (Bhattacharya & Getoor, 2007; Frontini, Brando, & Ganascia, 2015; Song, Luo, & Heflin, 2017; Zhu, Ghasemi-Gol, Szekely, Galstyan, & Knoblock, 2016). Collective approaches usually carry more scalability challenges compared with FBS systems. However, it has been proved that those algorithms could be adapted to be more scalable (Rastogi, Dalvi, & Garofalakis, 2011). In fact, a recent study has pointed out that 26.23% of the publications of record linkage techniques in Big Data environments over the past seven years rely on graph-based approaches or are thought to be applied to graphs of Linked Data (Enríquez et al., 2017).

One of the main tools used to improve the scalability of systems are the blocking /clustering techniques. When trying to find matches between entities of two databases A and B, assuming that every a_i in A is a possible match for every b_j in B would lead to a quadratic complexity hardly scalable. Many approaches were early suggested to reduce that complexity via filtering the number of potential matches, such as sorting of records in order to keep similar contents together (Hernández & Stolfo, 1995), clustering of candidates with computationally cheap functions before employing more expensive methods to compare potential pairs(Chaudhuri et al., 2003) or q-gram indexing (Baxter, Christen, & Churches, 2003). Current investigation lines are going deeper in the development of blocking techniques for large, heterogeneous and possibly semi-structured data (Efthymiou, Papadakis, Papastefanatos, Stefanidis, & Palpanas, 2017; Papadakis, Ioannou, Niederée, Palpanas, & Nejdl, 2012), as well as graph-like environments (de Assis Costa & de Oliveira, 2016; Fisher, Christen, Wang, & Rahm, 2015; Shin, Jung, Lee, & Kang, 2015).

Developing or designing a reconciliation system needs to include combinations of the algorithms mentioned and several other features. A great example could be privacy preserving matching (Vatsalan, Christen, & Verykios, 2013; Vatsalan, Sehili, Christen, & Rahm, 2017), handy when several organizations take part in the matching process but the information to be linked is sensible or should be encrypted. In addition, some extra challenges or decisions must be addressed, such as accepted input/output formats, mechanisms to interact with the user (API, library, web application, etc.), configuration options, the possibility of including extra algorithms/workflows.

We have explored a commercial patented system specialized in recognition of music metadata (Dunning, Kindig, Joshlin, & Archibald, 2011) and several well-known open-source approaches, including Dude (Draisbach & Naumann, 2010), D-Dupe (Kang, Getoor, Shneiderman, Bilgic, & Licamele, 2008), SILK (Volz, Bizer, Gaedke, & Kobilarov, 2009), BigMatch (Yancey, 2002), FEBRL (Peter Christen, 2008), FRIL (Jurczyk, Lu, Xiong, Cragan, & Correa, 2008), Merge ToolBox (Schnell, Bachteler, & Reiher, 2009), NADEEF/ER (Elmagarmid et al., 2014), and MusicBrainz Piccard (Stutzbach, 2011). We have checked the features of those systems and we provide a qualitative comparison of them with MERA architecture, which is shown in Table 1.

Despite the fact that all of them tackle the challenge of entity matching, they are really heterogeneous. Table 1 also shows that MERA is the most flexible and configurable system among the music-specialized ones. In addition, to the best of our knowledge, none of the existing systems include the following MERA's features:

- Integration of several sources to be used during the reconciliation process in a reusable graph, with a scheme which maintains which pieces of information have been provided by which source(s);
- Identification of alternative textual forms of an entity during the matching process via configurable complex paths in a graph.

Table 1. Comparison of record linkage systems

		MERA	BigMatch	FEBRL	SILK	DUDE	D-Dupe	FRIL	Merge ToolBox	NADEEF/ ER	CD Patent	MB Piccard
Main	Linkage	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
	Deduplication	-	✓	✓	-	✓	✓	-	✓	✓	-	-
Comparison algorithms	Include	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Let choose	✓	-	✓	✓	✓	✓	✓	✓	✓	-	-
	Let combine	✓	-	✓	✓	✓	✓	✓	-	✓	✓	✓
	Let implement	✓	-	✓	-	✓	✓	-	-	✓	-	-
Blocking techniques	Include	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Let choose	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-
	Let combine	✓	✓	✓	✓	✓	✓	-	-	-	✓	-
	Let implement	✓	✓	✓	-	✓	✓	-	-		-	-
Training data	Include	-	-	✓	-	-	-	✓	-	-	-	-
	Let choose	-	-	✓	-	-	-	✓	-	-	-	-
	Let combine	-	-	✓	-	-	-	-	-	-	-	-
	Let implement	-	-	✓	-	-	-	-	-	-	-	-
Collective Matching		✓	-	-	-	-	✓	-	-	-	-	✓
Privacy preserving		-	-	-	-	-	-	-	✓	-	-	-
GUI		-	-	✓	✓	-	✓	✓	✓	✓	✓	✓
Multi-source		✓	-	-	-	-	-	-	-	-	✓	-
Alternative entity forms		✓	-	-	-	-	-	-	-	-	-	✓
Music specialized		✓	-	-	-	-	-	-	-	-	✓	✓

3. SYSTEM ARCHITECTURE

MERA defines a strategy to combine several categories of algorithms or heuristics to conciliate two sets of entities $A = \{a_0, a_1 ... a_n\}$ and $B = \{b_0, b_1 \ldots b_m\}$, where the entities of B are stored in an RDF graph G. Each record a_i is processed independently to find its most similar records in B. The user of MERA should be allowed to define a set of conditions that an entity b_j must fulfill to be considered as a possible match. Consequently, every entity a_i could be associated with none, one or several results, depending on the number of entities $b_j \in B$ that fit in the conditions. In case of having more than one result for the same entity a_i, those will be ranked regarding its degree of similarity with the target element.

If we think of MERA as a black box, the expected inputs and the received outputs are the following:

- *Input1*: Set of entities $A = \{a_0, a_1 \ldots a_n\}$, for which every a_i will be processed independently;
- *Input2*: RDF graph containing the set of entities $B = \{b_0, b_1 \ldots b_m\}$;
- *Input3*: Configuration data, in which MERA's options are specified;
- *Output*: Association of each element a_i with the elements in B that fit in the matching conditions. In case of having several results for an entity a_i, they will be presented sorted in decreasing order by degree of similarity with a_i.

MERA has been designed to achieve better results with more prior knowledge of the nature of the sources to conciliate, so the user can configure several parameters of the matching process like:

- List of normalization algorithms;
- List of comparison algorithms;
- List of blocking functions;
- Relevance of related contents/entities when applying disambiguation;
- Thresholds of similarity that two entities should reach to be considered matches;
- List of sources (white/black lists) to use during the matching process;
- Maximum number of candidate results per query;
- Maximum number of candidates at the end of blocking stages;
- Relations between entities that should be considered;
- Alternative forms of entities that should be considered.

Configuration related to algorithms or acceptance values can be applied at different hierarchical levels, i.e., the user could indicate, for example, that Levenshtein distance should be applied in every case with the exception of canonical names, which will be compared using exact match. The user can also specify that the canonical name of albums will be compared with an exact match, but the canonical name of a group will use Jaccard similarity. In addition, MERA is designed to let a user implement new functions (normalization, comparison, blocking) that can be integrated into the base workflow.

The graph schema of MERA has been designed to let the user combine the knowledge of several sources during the matching process. Suppose a scenario in which we want to conciliate two sources A and B, where A contains civil names of artists and B contains artistic names. Also, we have access to a third source C which includes both civil and artistic names. In order to produce matches between A and B, we propose to turn B records in a graph G and link records of C with nodes of G. The obtained results allow us to build an enriched graph G' containing all B records in nodes decorated with information of C. By this, making queries with entities of A against G' will potentially improve the result of making queries directly against G. G' may be reusable for different purposes, even to use separately the information of B and C. Each piece of information is associated with its original source, and MERA purpose a way of just using the data from certain sources during the matching process.

3.1. Query Structure

Records in A are turned into queries that may involve one or more entities at a time. A query in MERA must specify a main type or main order o, a main content m, and a set R of typed refinements r, represented as pairs $r_i = (t_i, s_i)$. Let's consider a scenario in which we have a set A containing artists possibly associated to song titles. One of the artists of A is a_α = "Mike Chang", with no associated songs. Another artist is a_β = "Kurt Hummel", and he has associated the song s_β = "For Good". Now, let's consider that we want to conciliate the artists of A with a graph G. For that, we must turn entities in A into queries. The resulting query $q_\alpha = (o_\alpha, m_\alpha, R_\alpha = \{r_\alpha{*}\})$ for a_α would be ("artist", "Mike Chang", { λ }). The resulting query $q_\beta = (o_\beta, m_\beta, R_\beta = \{r_\beta{*}\})$ for a_β would be ("artist", "Kurt Hummel", {("song", "For Good")}).

As it can be seen, a MERA query q_i searches for a main raw string m_i associated to certain type o_i. Also, all the refinements or extra data provided for disambiguation purposes should also be labeled with a type.

3.2. MERA's Adaptable Algorithm

Algorithm 1 describes the MERA algorithm from the moment in which a Query q is received to the moment in which MERA returns its associated entities in Graph G. Table 2 contains the meaning of every macro used in Algorithm 1.

Algorithm 1. MERA query

```
Input: Graph G
Input: Query Q
Input: Config C
                                                      %% Blocking stage
candidateNodes ← G.nodes
for all fBLK ∊ C.blockingFunctions do
    candidateNodes ← $f_{BLK}$(q, candidateNodes )
                                                     %% Main Comparison
tmpResults ← [λ]
for all aNode ∊ candidateNodes do
    formScores ← {λ}
    for all aForm ∊ aNode.alternativeForms do
        formScores += fCMP (q.mainContent, aForm , q.type)
    if max(formScores) ≥ C.minMainScore(q.order) then
        tmpResults[aNode] = max(formScores)
                                                         %% Refinements
for all aRefType ∊ q.refinementTypes do
    for all aCandidateRes ∊ tmpResults do
        candidateRefs ← G.getRelated(aCandidateRes.node, aRefType)
        for all aRefinement ∊ q.refinementsOfType( aRefType) do
            formScores  ←  { λ }
            for all aForm ∊ candidateRefs.alternativeForms do
                formScores += $f_{CMP}$ ( aRef, aForm, aRefType)
            if max(formScores) ≥ C.minRefScore(q.order, aRefType) then
                refScore ←max(formScores) ·C.relevance(q.order, aRefType)
                tmpResults[aNode] += refScore
                                          %% Filtering and sorting results
results ←  [λ]
k ← C.maxResults(q.order)
```

```
for all candidate ∈ tmpResults do
    if candidate.score < $C$. minScore($q$.order) then
        results += candidate
sort(results)
if |results| ≤ $k$ then
    return results
else
    return results[0:$k$]
```

We have distinguished four parts in Algorithm 1:

- **Blocking Stage:** C stores a set of blocking functions to filter candidates of G. All those functions are applied over G nodes to obtain a subset of nodes which will be compared with q;
- **Main Comparison:** Query q has a main type o and a main string content m. m is compared with all the alternative forms of the candidates obtained in step 1;

Table 2. MERA query

Structures			
Model	**Meaning**		
{λ}	Empty set		
[λ]	Empty dictionary of pairs (key → value)		
Functions			
Name	**Parameters**	**Return**	
fBLK	Query *q*, set of nodes *S*	Subset *S'* of *S* with the best matching candidates for *q* in *S*	
fCMP	String s_1, string s_2, type *t*	Similarity score between s_1 and s_2, applying algorithms associated to type *t*	
sort	Set or dictionary *S*	Sort *S* in descending order	
Methods			
Name	**Parameters**	**Invoked over**	**Return**
minMainScore	Type of order *o*	Config *C*	Minimum score associated to *o* in *C*
getRelated	Node *n*, type *t*	Graph *G*	Nodes in *G* related with *n* with an edge of type *t*
refinementsOfType	Type *t*	Query *q*	Set of all the refinements in *q* of type *t*
minRefScore	Type of order *o*, type of entity *t*	Config *C*	Minimum score that an entity of type *t* should reach to be accepted when executing an order of type *o*.
relevance	Type of order *o*, type of entity *t*	Config *C*	Relevance factor of a refinement of type *t* when executing an order of type *o*.
Properties			
Name	**Invoked over**	**Return**	
nodes	Graph *G*	All nodes in *G*	
blockingFunctions	Config *C*	Set of blocking functions	
alternativeForms	Node *n*	Alternative forms of *n*	
mainContent	Query *q*	Main string in *q*	
type	Query *q*	Type of the main content in *q*	
order	Query *q*	Type of order in *q*	
refinementTypes	Query *q*	Set of types of refinements in *q*	

- **Refinements:** Query q may have a set of extra data or refinements. For each refinement r, G is navigated starting from the candidate nodes of step 1. If some coincidences good enough according to C are found, the obtained score is added weighted according to parameters in C to the respective candidate node score;
- **Filtering and Sorting Results:** The results associated with a query *q* are sorted in decreasing order. The user can define a maximum quantity *k* of results to be associated to a query *q*. If more results than *k* were found, only the *k* better are returned. Otherwise, all the results are returned.

3.3. String Comparison

In lines 11 and 20 of Algorithm 1, we use a macro f_{CMP} to represent the process in which the similarity between two strings *s* and *t* is obtained by using several comparison algorithms. f_{CMP} also receives a type *o*, expected to be the type of the entities that *s* and *t* represent. *o* must be specified for a potential accuracy profit, since different algorithms may be applied to conciliate *s* and *t* regarding their type *o*.

When mapping *s* and *t* to their degree of similarity *r*, once the set $P = \{P_0, P_1 \ldots P_k\}$ of preprocessing techniques and the set $C = \{C_0, C_1 \ldots Cl\}$ of comparison techniques has been selected based on the type *o*, the same workflow is followed in all cases, graphically described in Figure 1. *s* is turned into s_0 and *t* into t_0 pipping all algorithms P_i in *P*. That is, the input of P_0 is *s* and, in general terms, the input of P_i is the output of P_{i-1}. The final output s_0 is the output of P_k. Then, for each $C_i \in C$, we obtain every $R_i = C_i(s_0, t_0)$. The greater value of $\{R_0, R_1 \ldots R_i\}$ is taken as the result *r*, having $f_{CMP}(s, t, o) = r$.

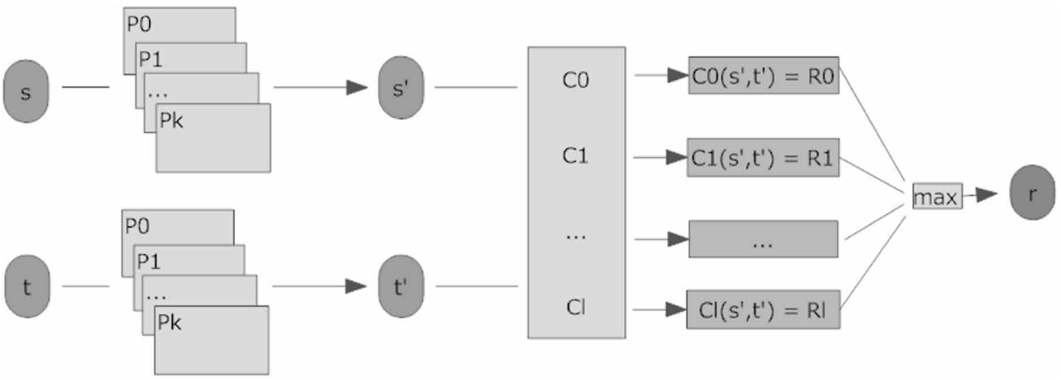
3.4. Blocking function

MERA may employ several blocking functions at a time. However, we also include a blocking function that manages the concept of alternative form of an entity, which consists of an adaptation of q-gram indexing and TF-IDF. The pseudo-code is included in Algorithm 2, and the macros are explained in Table 3.

Algorithm 2. MERA blocking

```
01: Input: Query $Q$
02: Input: Index I
03: $N \leftarrow$ extractUniqueQrams($q$.mainContent)
04: $E_{idf}$ = [$\lambda$]
05: $E_{ngr}$ = [$\lambda$]
```

Figure 1. Mapping two strings s and t into a real number r



```
06: for all $n_i \in N$ do
07:     $idf_{ni} \leftarrow I$.idf($n_i$)
08:     for all $e_i \in I$.entitiesWithQgram($n_i$) do
09:         $E_{idf}[e_i] \leftarrow E_{idf}[e_i]$+meraTfIdf($e_i, n_i, idf_{ni}, I$)
10:         $E_{ngr}[e_i] \leftarrow E_{ngr}[e_i] + 1$
11: result ← {λ}
12: for all $e_i \in E_{ngr}$ do
13:     if $E_{ngr}[e_i] = |N|$ then
14:         result ← result + $e_i$
15: return result ⋃ bestK($E_{idf}$)
```

The differences between the Algorithm 2 and classical TF-IDF approaches based in q-grams are:

- We introduce the concept of alternative form of an entity. An entity may be considered as a set of forms, and the algorithm takes care about how many forms contain a q-gram, instead of how many total times a q-gram is contained in all the entity forms;
- We manage the relevance of the q-grams in an entity with an accumulated TF-IDF score but also the number of q-grams contained in an entity *e*. If a certain entity contains all the q-grams of *q*.mainContent, then that entity is automatically included in the result set of candidates.

Table 3. MERA blocking

Structures			
Model	**Meaning**		
{λ}	Empty set		
[λ]	Empty dictionary of pairs (key → value)		
Functions			
Name	**Parameters**	**Return**	
extractUniqueQgrams	String *s*	Set containing all the q-grams in *s*	
meraTfIdf	entity ID *ei*,, q-gram *ni*, IDF score *idfni*, Index *I*.	*r* = *tf* · *idfni*, where *tf* is the amount of forms of *ei* in which the q-gram *ni* appears	
bestK	Dictionary *D*	Set containing *k* keys of pairs in *D* that are pointing to greater values. The parameter *k* is specified through configuration.	
Methods			
Name	**Parameters**	**Invoked over**	**Return**
idf	q-gram *ni*	Index *I*	Idf score of *ni* in *I*
entitiesWithNgram:	q-gram *ni*	Index *I*	Id of the entities in *I* that contain *ni*
Properties			
Name	**Invoked over**	**Return**	
nodes	Graph *G*	All nodes in *G*	
blockingFunctions	Config *C*	Set of blocking functions	
alternativeForms	Node *n*	Alternative forms of *n*	
mainContent	Query *q*	Main string in *q*	
type	Query *q*	Type of the main content in *q*	
order	Query *q*	Type of order in *q*	
refinementTypes	Query *q*	Set of types of refinements in *q*	

3.5. Graph navigation

MERA is designed to match queries with graph nodes that represent entities. Finding a result requires the use of string similarity measures and graph navigation through the nodes' neighborhood. MERA uses two types of graph navigation when resolving a query: looking for alternative forms and looking for related entities:

- **Looking for alternative forms:** An entity (node) e may be associated with several identifying or pseudo-identifying forms in a graph, reachable from e through different properties or complex paths. E.g., a node that represents an artist may be identified with properties such as canonical name, civil name, alias, or complex paths as all the identifying forms of the groups in which he participates;
- **Looking for related entities:** If the received queries include some refinement regarding other entities (CD of a track, writer of a song, partner of an artist, etc.) the graph will be explored looking for coincidences in that kind of relations.

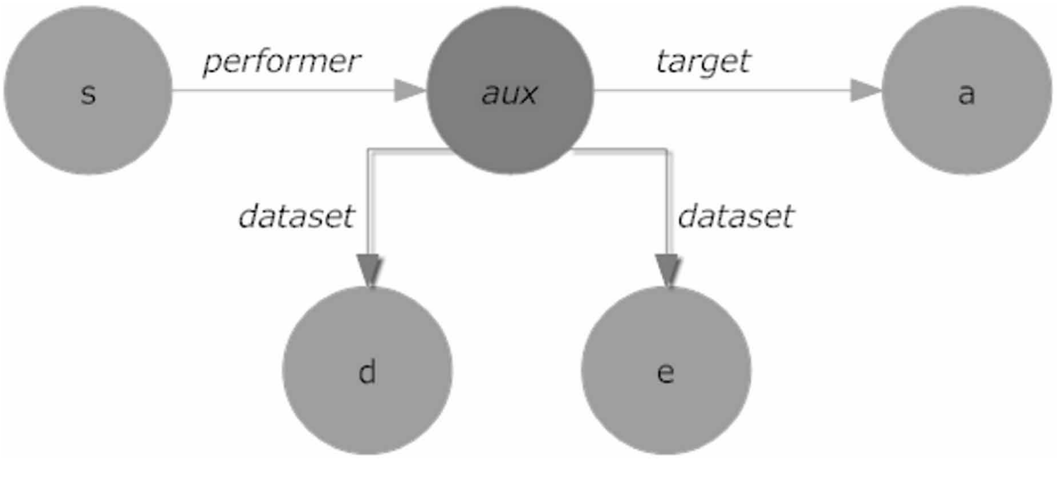
3.6. Graph Schema

MERA is designed to be able to manage different sources, storing the origin of each piece of information in a single Graph *G*. The strategy employed is reification, where the nodes are not directly linked with their related contents (other entities of string properties), but auxiliary nodes are used as intermediaries. Those auxiliary nodes relate the subject and the object while they point to a third node that represents the origin of the information.

For example, if we have a song *s* performed by an artist *a*. A natural way of representing this with an RDF graph is the triple $t = (s,$ "performer", $a)$. However, we also want to keep the information of *according to who s* is performed by *a*. Suppose that this information has been obtained from a dataset *d*. In general terms, a triple $t = (s, p, o)$ that has been obtained from a dataset *d* is transformed in three triples (s, p, n_{aux}), $(n_{aux}, target, o)$ and $(n_{aux}, dataset, d)$.

Let's say that we want to add the info of a new dataset denoted as *e* to graph *G* and, according to *e*, song *s* also has *a* as a performer. The resulting MERA graph is the one shown in Figure 2. The relation we want to represent is the same as in the previous case, i.e., the triple (*s*, "performer", *a*), but there is an already existing auxiliary node *aux* representing that link. Then, the triple (*aux*, "dataset", *e*) in *G* to indicate that dataset *e* also agrees about that link between *s* and *a*.

Figure 2. Auxiliary node with several datasets



4. EVALUATION

We have implemented a prototype of MERA which includes the following characteristics:

- Graph navigation exploring alternative forms for each entity;
- Graph navigation exploring related entities;
- Configuration of relevancies when applying refinements to a query;
- Configuration of minimum acceptable values for each type when scoring a result;
- Blocking function adapted to alternative entity forms as described in section 3.4;
- Usage of MERA RDF graph schema;
- Set of comparison algorithms of general-purpose;
- Set of text standardization functions.

The prototype has been used to conduct two kinds of experiments: On the one hand, intra-comparisons tests to evaluate which is the influence of the different features of MERA over the matching results. On the other hand, comparison of the results obtained by our prototype with the results of a baseline system matching the same sources.

Our experiments are all based on the reconciliation of a random slice of two different musical sources *A* and *B* against a third source *C*. The information of *C* has been introduced in a graph *G* and the entries of *A* and *B* have been used to build queries. The expected result for each query has been manually determined and annotated, which has allowed us to measure the correctness of the results automatically obtained by MERA and the alternative approach. Source *A* is supposed to contain data of high quality (complete and with none or few misspellings) while source *B* is supposed to contain noisy data (user-entered and incomplete). However, every query thrown against *G* is formed by a song name and one or more names of associated artists/writers. The selected sources are:

- **Source A of high quality:** MusicBrainz (Swartz, 2002);
- **Source *B* of noisy data:** Queries extracted from AOL dump (Arrington, 2006);
- **Target source *C*:** Discogs (Hartnett, 2015).

Some entries were randomly selected from the material of MusicBrainz and AOL to build queries. The same queries were used during all the experiments. Since the targeted source is the content of Discogs, the advanced search engine of Discogs itself has been used to compare with MERA. As well as our proposal, Discogs advanced search engine allows the user to perform rich queries in which every piece of information is labeled with a type of the musical domain (release, artist, writer, genre…).

4.1. Description of Experiments

The experiments were executed under the following software/hardware conditions: Virtual Machine using Windows Azure cloud computing services, 64-bit Operating System: Windows Server 2012 R2, AMD Opteron (tm) Processor 4171 HE 2.10 GHz, 14 GB RAM and Python 2.7.3 64-bit.

4.1.1. Target Source: Discogs

We have used Discogs (Hartnett, 2015) to create a graph *G* containing a total of 500000 songs, as well as their associated artists and writers. The information has been extracted from a dump of Discogs releases published in 2015-01-01. That dump includes a set of musical releases containing one or more tracks and associated artist, with a total of 45458287 detected tracks among all the releases.

The scalability of our prototype is not enough to manage a graph with the data of the entire dump, since it completely works in main memory, so we have reduced that quantity to 500000, a number that we can handle but that is still a representative and a high enough slice. Those 500000 songs include 205 that has been manually detected to be the adequate answer to the queries used in the experiments. The rest of songs has been randomly selected from the dump file.

We also included in the graph all the artists and writers linked to each song. Discogs associates people to tracks/releases using different roles. We have mapped some of the original Discogs roles to our own roles to include these associated people in our graph. Discogs roles "Artist", "Featuring", and "Vocals" have been mapped to "artist", and "Written-By" has been mapped to "writer". The rest of Discogs roles have been ignored.

The final graph used in the experiments contains a total of 500000 songs and 624441 people.

4.1.2. Source of High Quality: MusicBrainz

The open music encyclopedia MusicBrainz (Hemerly, 2011) fits in our requirements of offering a great amount of data of high quality. We have randomly selected a set of works from this source and we have manually checked that they point to a work that also exists in Discogs data dump. In order to generate a sample of MusicBrainz, the entire database was downloaded and 300 recordings were randomly picked. Later, we manually looked for coincidences between these elements and song nodes in the graph formed by the partial content of Discogs. The first 100 coincidences were selected and used to build queries using the song title and the artists/writers associated to them in MusicBrainz. The average of associated artist/writers per song is 2.62.

4.1.3. Noisy Source: AOL Dump

In 2006, AOL Inc. released a file containing twenty million search keywords for over 650.000 users over a three-month period (Arrington, 2006). We have considered that musical-related searches found among this material could be a representative example of a noisy source, since all the inaccuracies have been introduced by random real users when doing real searches. The strategy to find musical content among the rest of searches has followed the next steps:

- We explored all the items selecting those that contained the tokens "feat" and "featuring". We divided those items into two parts using the word "feat" as separator, and we added each one to a set of noisy candidate keys *N*;
- We removed from *N* those words or sequences that are meaningless for our purposes, such as "to download", "ringtone", "free music", "song of", "video", etc. In case of finding "-" in a key, we divided the key into two pieces and we repeated the cleaning process with each part. All the results were added to a set of clean candidate keys *C* with 1203 elements;
- We revised manually each key in *C* in the context of its apparition in the log of AOL to check if they really were artists/songs or not. In addition, if we detected that some of the keys contained more than one artist/song at a time, we divided it in parts and considered it separately. At the end of this process 922 sequences (not unique) representing songs or artists were detected. We stored them in a set of found keys *F*;
- We put all the elements of *F* in a set of definitive keys *D*. Then, for every key k_i in *F* with a length greater than 3, we generated all the possible strings $S_i = \{s_0, s_1 \dots s_k\}$ in a Levenshtein distance of one with k_i. For each $s_j \in S_i$, if s_j was found more than 5 times in the log of AOL, it was added to *D*. With this, we collected a group of detected musical entities that were queried by users as well as character variations of all of them that had a certain presence in the log of AOL. We removed from *D* meaningless sequences as the ones used in previous steps;
- We processed again the entire log looking for searches containing at least a key of *D*. We elaborated three different lists with the found searches, regarding if they contained a single key, between two and three or more than three. 327,904 searches were detected;
- We messed those lists randomly. Then, we revised manually the lists to extract the first 35 results of each one that we could identify as searches for songs that included, at least, one artist name. We erased all the meaningless words or sequences of previous steps from the results.

In every case, we respected the original appearance of each entity in the log of AOL. That is, we identified different parts in each element and we removed meaningless words, but no other changes were applied to the original content.

With this, we obtained a list of 105 items with material to build queries with noise introduced by real users. The average of associated artists per song is 1.44.

4.1.4. Measuring the Effect of MERA Features

Some experiments designed to check the impact of MERA's proposals over scenarios with different level of data quality were executed. The tests evaluate the correctness of the results returned by our prototype with different configurations when executing a consistent set of queries against the very same source. In all cases, the entities to recognize were songs, and the information used was its title and associated artists. The sets of chosen features across the different experiments are shown in Table 4.

There are some other configuration parameters whose values have been fixed for all the experiments, including:

- Top results per query: 15
- Minimum score for a song to be accepted: 0.50
- Minimum score for an artist to be accepted: 0.65
- Artist relevance when refining the result of a song: 0.80
- Reconciliation string algorithms used in all cases: Levenshtein similarity
- Pre-processing functions included: replacement of all non-ASCII characters by ASCII equivalents with Unicode normalization, lower-casing, deletion of punctuation marks, and deletion of redundant white spaces.

In absence of the MERA blocking function described in the Algorithm 2, the blocking strategy consisted on selecting the best 60 candidates based on accumulated TF-IDF score of q-grams. In case of being active MERA blocking, the candidates may exceed that number if there are enough individuals containing all the q-grams of the song provided in the query. The length of the q-grams was, in all cases, 3 characters.

This configuration allows relatively bad results to be ranked (at a low position) as possible matches, mainly because of the low threshold of acceptance for songs. An execution of MERA looking for just safe matches should probably define higher values of acceptance. However, since we are purely evaluating the correctness of the results, in this case we prefer to prioritize exhaustiveness by sacrificing performance (the more accepted songs, the more comparisons will be performed). Needless to say, the usage of Levenshtein distance as the unique string similarity may cause false negatives when trying to match entities which are named using disordered tokens or abbreviations not recognized as alternative forms.

Table 4. Features used in experiments

Feature	Experiment ID				
	A	B	C	D	E
Considering alternative forms of entities	-	✓	-	✓	✓
Using information of related entities	-	-	✓	✓	✓
Using MERA blocking function	-	-	-	-	✓

The alternative forms considered for each entity when building the graph *G* were the following:

- **Artists:** Canonical name, civil name, alias and usual name variations such as abbreviations. All this information is provided by Discogs. When dealing with a group, also the canonical name of its integrating artists. When dealing with a person, the canonical name of the groups she belongs to;
- **Songs:** Canonical name. When this name has text between brackets (e.g., "Summer love (radio remix)"), also the same name without the information in brackets was considered as an alternative form.

4.1.5. Comparing MERA with Discogs Advanced Search

Among the available alternative matching systems mentioned in section 2, Discogs advanced search system[1] was selected as baseline mainly because of two reasons. On the one hand, in our experiments we aim to match entities of different sources with the content of Discogs. Then, it seems like the search engine of Discogs itself may be an appropriate tool to consider for such a goal. On the other hand, the input information expected by the advanced search system of Discogs is pretty similar to the conceptual information expected by MERA. Most of the alternatives mentioned in section 2 are general-purpose and do not accept a labeled input such as MERA expects. Despite it may be possible to adapt some of those systems to build conditions similar to our experiments in terms of targeted sources and type of queries, probably the closest systems to our design are the music specialized ones, such as Discogs search engine or MusicBrainz Piccard. Nevertheless, the former is designed to match entries with MusicBrainz database.

Discogs portal also offers a type of unlabeled search as a regular general-purpose search engine. However, this type of search is hardly comparable to MERA, especially when dealing with queries which have extra associated information further than the name of the core element to match. Because of this, only the advanced search system has been compared with our prototype.

As far as we know, Discogs has not released software separate from its web application to perform queries against its library. Due to that, our evaluation has consisted in introducing manually the contents of each query in the web form of advanced search and annotating the obtained results. Each query has been thrown using different slices of the whole information to check how the presence/absence of data affects the results. For instance, given a query to match a song called "Heart-Shaped Box", with "Nirvana" as artist and "Kurt Cobain" as writer, the results of using just the of the name of the song in the form, song plus artists, song plus writer, and all the information at a time have been annotated.

All the queries from both target sources were introduced in the Discogs system. In the case of AOL, which contains information already typed, artists were specified in the field with the label "By Artist". The information related to writers have been put into the field "Credit", which points to every other role different to "artist". However, in the case of AOL source, the content was typed a posteriori, then it is not clear if the user who originally wrote the query was referring to an artist, a writer or to any other role. Due to that reason, the information of people was introduced in both fields of the form, and just the best-obtained result was annotated.

As we previously stated, the current version of our prototype operates just in main memory and cannot handle a graph containing the entire dump of Discogs. Nevertheless, the Discogs web search system always uses the entire information of the source. In order to produce comparable results, when we threw a query and the desired entry did not appear in the first position of the results, we checked each element which ranked better to determine if it were also included in our random slice of Discogs. All those entities that were not in our slice were excluded from the obtained list, and the expected entry was promoted to its corresponding rank without these elements. This decision could have affected the quality of the experiments if Discogs applied a blocking function with a maximum number of candidates, such as MERA does. However, it does not seem that there is a maximum number of results when using the web search.

4.2. Results and Discussion

4.2.1. Inclusion of MERA Features

Experiment A consisted of applying Levenshtein distance to conciliate sources without using any kind of graph navigation, i.e., just by comparing canonical titles of songs. The obtained results can be queried in Figure 3. In the chart, we have grouped the results of the queries in four different categories:

- Queries in which the correct entry appeared in first place;
- Queries in which the correct entry appeared in second or third place;
- Queries in which the correct entry appeared in fourth or worse place;
- Queries in which the correct entry did not even appear.

Under these conditions, 52% of MB's target results are ranked first. In AOL case, just 43.81%. The reasons we have found for these low rates of correct are mainly the same for both sources:

- **The blocking function did not detect as candidate the target entry:** This happens mostly when dealing with songs with a short name, with common words in *G*, or a combination of these two factors;
- **Lack of alternative forms of song:** The titles of AOL and MB may present a great difference with their corresponding title in Discogs according to Levenshtein distance, despite they point to the same reality. This is mainly because Discogs' content usually contains song titles with information between brackets, such as "Summer love (radio edition)" or "Summer love (remix DJ)";
- **Ambiguity:** Song names are not unique in some cases. If we do not explore related artist in order to decide the best result, there is not a reliable criterion to decide which entry should appear in first place.

The conditions of experiments B and C allow us to solve issues derived from lack of alternative forms of songs and ambiguity respectively. These results are respectively shown in Figure 4 and Figure 5. Note that the inclusion of MERA's graph navigation strategies improve the results for both sources, increasing the rate of target entities ranked first and decreasing the rate of intermediate/missed entries. The inclusion of related entities in Experiment C produces a greater positive impact on the results, improving the rate of first-ranked entities. Nevertheless, it has poor or even none impact in the rate of missed entries. This happens because most of the missed results are not accepted in the blocking stage. MERA explores relations between entities that have passed the blocking cut, so intermediate results can improve their rank, but this does not affect at all to those that have not

Figure 3. Experiment A: No graph navigation

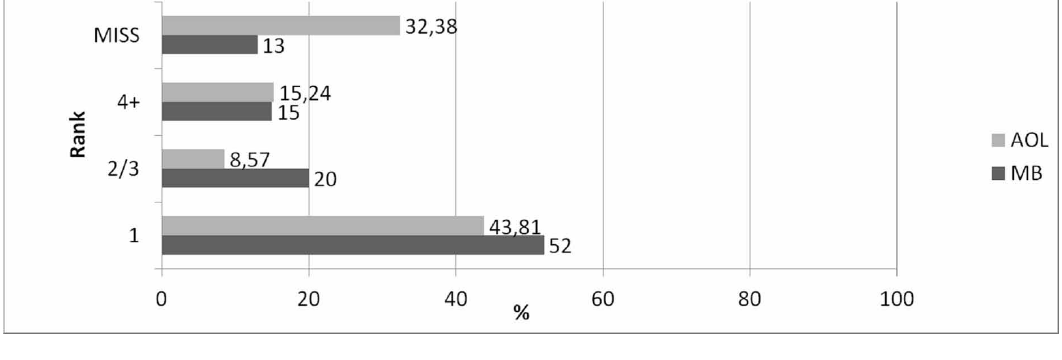


Figure 4. Experiment B: Alternative forms

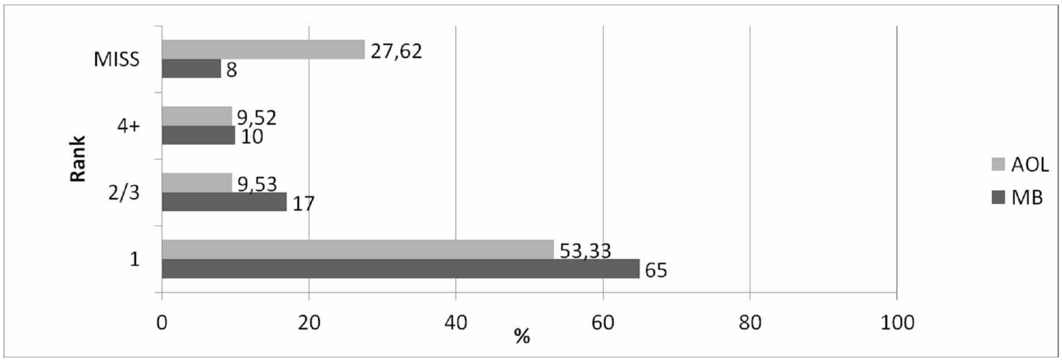
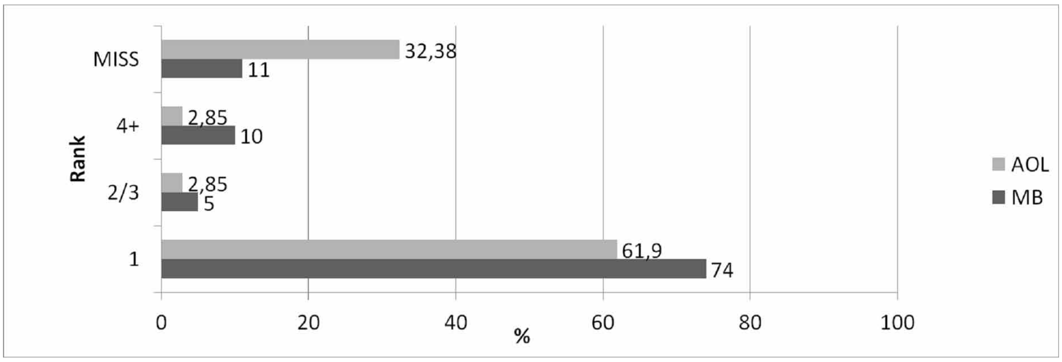


Figure 5. Experiment C: Related entities

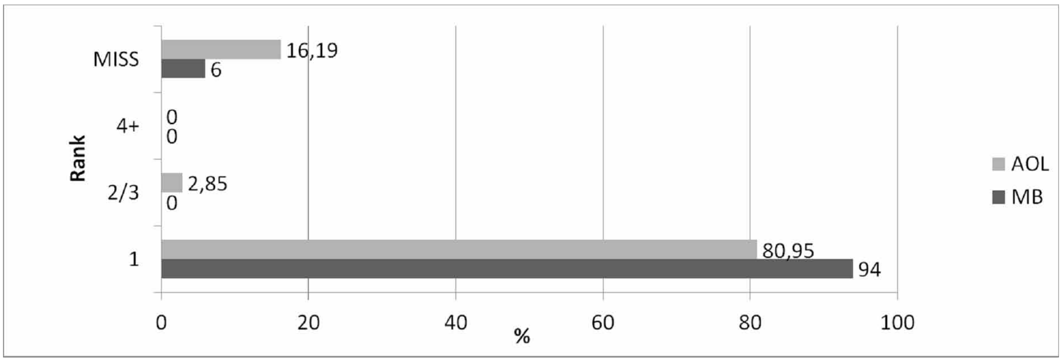


reached the comparison stage. On the other hand, in Experiment B we have added alternative forms that affect the TF-IDF of some target entities, which allow some missed results to pass the threshold during the blocking stage and let them be evaluated by the comparison algorithms.

In Experiment D, we have applied both graph navigations at a time, i.e., we consider alternative forms and related entities during the matching process. This has a positive effect on the success rates for both sources. The results can be checked in Figure 6. In this case, the reasons to find a target entry missed or low-ranked are not the same for both sources:

- **MusicBrainz:** There are 6 target entries not ranked first. One of them did not pass the blocking function threshold because the song has a short name formed by common q-grams in Spanish: "El amor". The other five are all pieces of classical music in which it looks like Levenshtein distance cannot detect matches. An illustrative example of this is the query "Sonata for Piano no. 7 in C major, K. 284b/309: III. Rondeau. Allegretto grazioso". Its equivalent in Discogs express the same reality but with different naming conventions. Artist information has not been helpful in these cases since the classical composers provided by MusicBrainz for those pieces (Mozart, Beethoven...) are widely repeated across other tracks with similar names. Algorithms different to Levenshtein should be applied to match this kind of works;
- **AOL:** The under-ranked entries happen mainly because of the detection of versions of songs before the one we are looking for. The reason for the missed ones is, in most of the cases, that the target entry did not pass the blocking function threshold. Some results are still missed because

Figure 6. Experiment D: Complete graph navigation

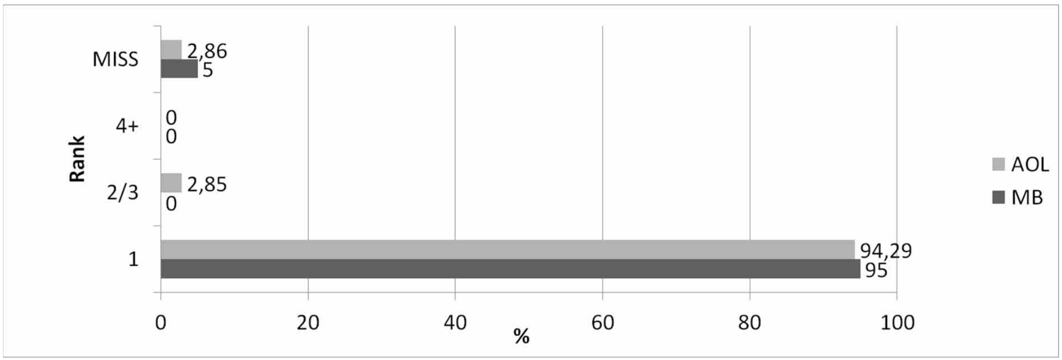


the song name contained hard misspellings. An illustrative example is the string "ghostrider" trying to express the song "Ghost Writer".

When we include MERA blocking function in Experiment E, all the target results pass the blocking cut and the rate of first-ranked entries grows to 95% for MB and 94.29% for AOL. The under-ranked results are the same of Experiment E, since we have not change comparison techniques nor we used extra data. The obtained results are shown in Figure 7.

Nevertheless, this blocking function has a performance cost. We repeated 50 times the experiments E and D and our measurements indicate that the performance is 14.76% worse in experiment E compared with experiment D. This cost is not just because of the calculations needed to select the candidates in blocking stage, but because that approach may generate more candidates than the pre-configured top number. This leads to more operations during the comparison stage. We have measured how MERA blocking function exceeds the preconfigured number of candidates to be accepted in blocking stage in experiment E. That number had been set to 60, and we checked that the average number of candidates returned per entity has been 68.12. Also, 94.62% of the queries had exactly 60 results. However, we have detected that in some cases the function had returned an amount of candidates that exceed by large 60. This had happened with queries formed by short names with common q-grams in *G*. While executing E, the query that produced a higher number of valid candidates in blocking stage has been "So what" of the artist "Ciara", with a total of 326 detected candidates.

Figure 7. Experiment E: Using MERA blocking function



4.2.2. Data Quality Effect with MERA

As shown in Figure 3, Figure 4, Figure 5, and Figure 6, if we do not use all the MERA features implemented in our prototype, the results of the clean source (MB) improve the results of the noisy source (AOL). However, when using all them at a time, as it is shown in Figure 7, both sources present very similar results. Actually, AOL obtained a lower rate of missed songs (2.86% vs 5% of MB). Nevertheless, these measurements may be conditioned by the size and nature of the samples. More tests are needed to measure how data quality impacts over results when using MERA.

4.2.3. MERA vs Discogs System

During the evaluation of MERA features in the previous experiments, we could check that the inclusion of an extra feature has a positive effect on the obtained results. This means that, in general terms, the more information is associated with a query, the more accurate are the results brought by MERA. However, Discogs search engine does not behave equally. The algorithms used by Discogs look for entries in which all the information introduced is included somehow. This may cause false negatives under the following conditions:

- When the information of the query exceeds the information of the target entity;
- When some of the elements of the query do not produce a match with some of the elements of the target entity due to misspellings or alternative names.

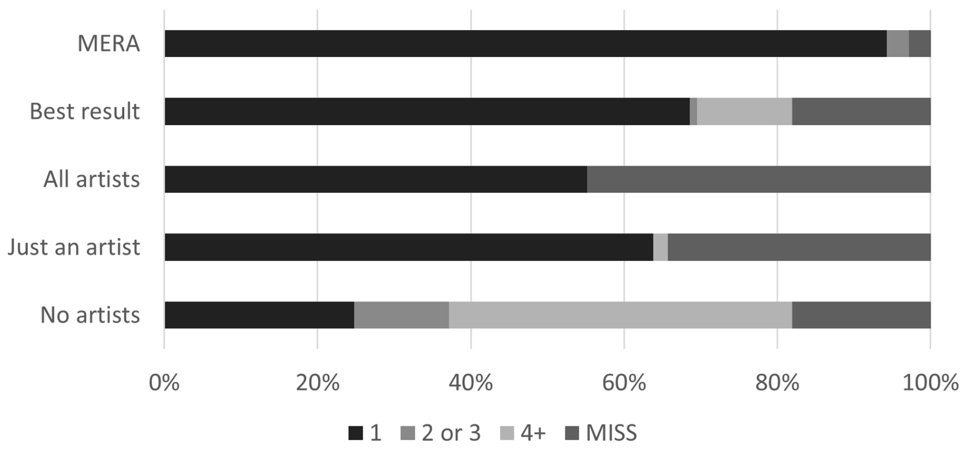
Due to that reason, using all the available information is not always the best approach to find the correct result with Discogs search engine. We have detected that 46.81% of the target entries using AOL material ranked worse with the inclusion in the query of some extra piece of information. In the case of MusicBrainz, this rate grows up to 64%.

Bearing that in mind, we present the results of our experiments using Discogs search engine segregated by the number and nature of the pieces of information used in each attempt to match an element of AOL/MusicBrainz with an entry in Discogs. We analyze separately the material of AOL and MusicBrainz.

4.2.3.1. AOL Against Discogs and MERA

In Figure 8 the results of using different strategies to match the very same element of the AOL material with its corresponding entry in Discogs are offered. Those strategies are identified with a

Figure 8. AOL against Discogs and MERA



label in the Y axis of the chart. Each label has the following meaning (in every case at least the song name was included in the query):

- **MERA:** MERA results using its complete set of features. These statistics are the ones obtained in Experiment E;
- **Best Result:** Best obtained result using any of the previous strategies for each entry;
- **All Artist:** All the available artist, in case there are two or more;
- **Just an Artist:** The associated artist who produced better results;
- **No Artist:** Just a song name.

Not all the queries fit in all the strategies. In the "All artists" section, just those entries with two or more associated people are considered. In every case, the results are relative to the number of cases which fit in the strategy.

Several conclusions can be extracted from the obtained data. Firstly, it seems like the best strategy to find the target entity for each query is using a single artist as extra information. The rate of queries which are resolved with the desired entry in the first position is 63.81% with just an artist, compared with the 55.17% obtained for all artists and the 24.76% with no extra information rather than song name. The reason behind this is the general strategy followed by Discogs search engine, in which all the information introduced in the search form must be part of an entry to be included in the results. Then, using an artist in the query usually works as a useful tool to decide the best entry among several songs with the same or similar name. However, including a piece of information which is not stored in Discogs cause a false negative.

As well as MERA, Discogs produced poor results of desired entities ranked in the first place when using just song names. Just 24.76% of the entities are successfully matched without providing artists. With MERA this number grows up to 43.81% without using alternative forms, as it is shown in Figure 3 (Experiment A). Despite MERA's rate is sensibly higher, none of the systems are accurate enough to consider that they produce good enough matches at the first attempt. Nevertheless, Discogs outperforms MERA in the rate of missed results using just a song name. 32.38% are missed with MERA, by 18.10% that are missed with Discogs. This difference is mainly caused because of the effects of the configured blocking function in Experiment A. The desired results missed MERA's cut, while they are simply low-ranked using Discogs.

Independently of the information combination employed, MERA outperforms Discogs effectiveness. As it is shown in Figure 7 (Experiment E), where all the features of our prototype were active, MERA finds the desired result in 94.29% of the cases, compared with the 68.57% achieved by Discogs. The reason behind this is the different text reconciliation algorithms used. Discogs search engine, which is based on ElasticSearch (Hartnett, 2015), do not follow a strategy of exact match, since it is capable of producing matches between titles with some missed or disordered tokens. However, it is heavily affected by misspellings or any other internal change in a token. An example of this is the query whose expected match is the song *Fine Young*, of *Cannibals*. The input found in the dump of AOL for this recording was "fine young", of "canibals". A query which uses just the song name is resolved with the desired entry ranking first. However, when introducing "canibals" in the field of artist, no results are offered by the search engine because of a missing "n" in the group name. Another significant example is the query "'touch it' remix", with single quotes surrounding "touch it" and no blank between the final quote and "remix". The absence of a blank after the quote cause that no result is shown. Just by adding that white space to the original query, the first result obtained is the target one.

4.2.3.2. MusicBrainz Against Discogs and MERA

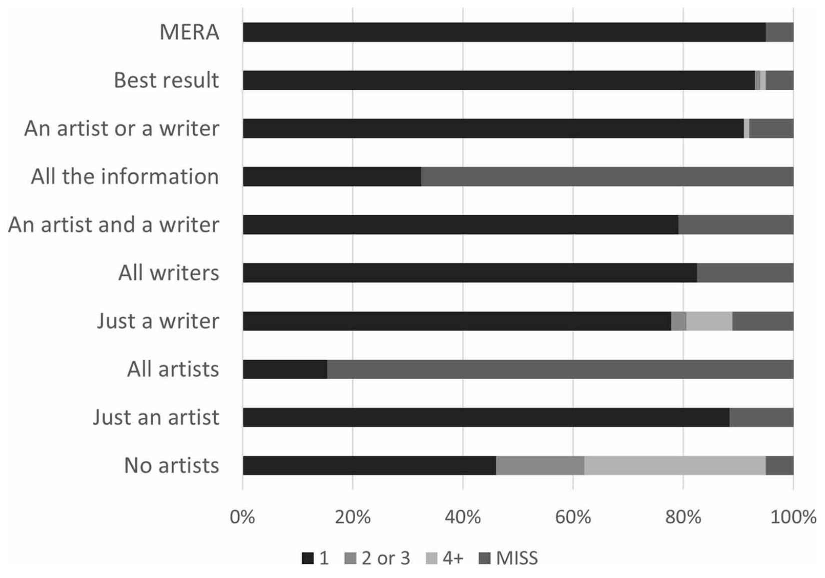
The results of using different sets of information of an entry of MusicBrainz to match it with its equivalent in Discogs using the web search engine are shown in Figure 9. Due to the nature of MusicBrainz information, it has been possible to test more combinations than in the case of AOL. All the people related to a song in MusicBrainz has an explicit role (main artist, writer, featurer, etc.). In this case, it is clear where each piece of information should be placed in the search form, in opposition to the case of AOL were the roles of related people were a priori unknown. This lets us analyze in a separate way the effect of including pieces of information of different types in the form.

The different strategies are identified with a label in the Y axes. The labels have the following meaning (in every case at least the song name was included in the query):

- **MERA:** MERA results using its complete set of features. These statistics are the ones obtained in Experiment E;
- **Best Result:** For each entry in MusicBrainz, best result achieved by using any of the previous strategies;
- **An Artist or a Writer:** The name of the artist or the writer who, by just himself, produced the best result;
- **All the Information:** The name of every artist and every writer, just in case this number of related entities is greater or equal to three;
- **An Artist and a Writer:** The name of the artist and the writer which produce a better result, just in case there are at least an artist and a writer;
- **All Writers:** The name of all the writers at a time;
- **Just a Writer:** Just the name of the writer who produced the best result;
- **All Artists:** All the names of all the artists at a time;
- **Just an Artist:** Just the name of the artist who produced the best result;
- **No Artists:** Just a song name.

As in the case of AOL, not all the MusicBrainz entries fit in all the strategies. In every case, the offered results are relative to the number of cases which fits in the strategy.

Figure 9. MusicBrainz against Discogs and MERA



One of the most relevant facts that can be extracted from our results is that, with a clean source and an adequate combination of artists and writers, the effectiveness of MERA and Discogs are quite similar. With all its features activated, our prototype was able to rank first the adequate entity 95% of the times, compared with the 93% scored by Discogs. Nevertheless, it is noticeable that the score of Discogs is obtained through considering the best attempt for each entry of MusicBrainz, instead of using a consistent strategy. Using a consistent strategy, the inclusion of a single artist in the query is the best approach to produce perfect matchings, with a success rate of 88.41%.

The results of using all the available information in a query may look shocking a priori: just 32.43% of the entries with several artists and writers associated produced a match, while 67.57% of the attempts did not return the target entity among the results (actually, all the results consisted of an empty list in those cases). This occurs because of using too much information. As we have already mentioned, a query with data more complete than the corresponding entry in Discogs cause a false negative.

Note that the inclusion of an artist, with a success rate of 88.41%, has a more positive effect on the results than the inclusion of a writer, with a success rate of 77.78%. The reason for this result is that it is common to have songs versioned by different artists. In a well-documented source, such as Discogs, all these versions are connected to the same writer. Then, the search engine is capable of finding all these entries, but it does not have extra information to decide which of these versions is the target one. On the contrary, specifying the artists of a given song works better as a disambiguation mechanism.

Although MERA and Discogs have the same rate of missing results for AOL queries (5%), the reasons for failing are not the same for both approaches due to the different capabilities of the algorithms used. As we already mentioned, MERA failed to recognize pieces of classical music with long names and disordered tokens. This may be solved by integrating into MERA's pipeline a token-oriented algorithm such as Jaccard similarity. Further research can be done to check the effect on false positives due to that decision.

By contrast, Discogs is able to properly match some of these titles of classical music, but it fails with other entries due mainly to slight differences in song or artists' names. An insightful example of this is the query "Cholly (Funk Gettin'Ready to Roll)". The corresponding entry in Discogs for that query is stored with a blank after the single quote. The absence of a blank made the search engine to consider "Gettin'Ready" as a single token, which is not present in any title of the source, which makes the application to return an empty list of results. An example of a query which is solved with the desired entry under-ranked is "Feel Like Makin'Love", written by "Gene McDaniels". In this case, the absence of blank between "Makin" and "Love" does not motivate the failure, but the way of specifying the author of the song. Discogs advanced search system considers alternative forms of artists, such as MERA does. When matching clean sources such as Discogs and MusicBrainz, an exact match with some of that forms is enough most of the times to recognize an artist. For instance, this writer is stored in the system with the forms of "Eugene McDaniels", "E. McDaniels" or just "McDaniels". However, in this case, the form "Gene McDaniels" used in MusicBrainz do not appear as a recognized alternative form, which causes that the correct entry appears under-ranked in the result list.

5. CONCLUSION AND FUTURE WORK

We have described MERA, the architecture of a highly configurable system for matching several data sources containing musical metadata. The main contributions of our approach are:

- Graph-based approach in which every piece of information maintains a list of source(s) that agrees on it, allowing the user to use this information for filtering contents;
- A configurable graph navigation system that looks for alternatives identifying forms of each node to use in the matching process.

We have also proposed a blocking technique based on q-gram indexing and TF-IDF adapted to MERA's notion of alternative forms of an entity. We implemented a prototype including these features and we found that it was effective in more than 94% of the cases under the conditions of our experiments, which included sources with different levels of data quality. We have compared our prototype with a well-known search engine music-specialized, outperforming its rate of correct matches in the two experiments that we developed.

As future work, we plan to improve MERA architecture to produce a system which includes the following features:

- Machine-learning algorithms that can infer adequate configurations through training data;
- Improve the scalability and efficiency of graph storage, management and access;
- Implementation of a scalable framework with parallel computing possibilities during the comparison stage.

We will also explore MERA's approach in a different scope to perform some test comparing our system with other general purpose approaches.

REFERENCES

Ananthakrishna, R., Chaudhuri, S., & Ganti, V. (2002). Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the 28th international conference on Very Large Data Bases* (pp. 586–597). doi:10.1016/B978-155860869-6/50058-5

Arrington, M. (2006). AOL proudly releases massive amounts of private data.

Baxter, R., Christen, P., & Churches, T. (2003). A comparison of fast blocking methods for record linkage. In ACM SIGKDD (Vol. 3, pp. 25–27).

Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S. E., & Widom, J. (2009). Swoosh: a generic approach to entity resolution. *The VLDB Journal - The International Journal on Very Large Data Bases, 18*(1), 255–276.

Bhattacharya, I., & Getoor, L. (2007). Collective entity resolution in relational data. [TKDD]. *ACM Transactions on Knowledge Discovery from Data*, *1*(1), 5, es. doi:10.1145/1217299.1217304

Bilenko, M., & Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 39–48). doi:10.1145/956750.956759

Chaudhuri, S., Ganjam, K., Ganti, V., & Motwani, R. (2003). Robust and efficient fuzzy match for online data cleaning. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data* (pp. 313–324). doi:10.1145/872757.872796

Christen, P. (2008). Febrl-: an open source data cleaning, deduplication and record linkage system with a graphical user interface. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1065–1068). doi:10.1145/1401890.1402020

Christen, P. (2012). *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer. Retrieved from https://books.google.es/books?id=LZrT6eWf9NMC

Cohen, W. W., Ravikumar, P., & Fienberg, S. (2003). A comparison of string metrics for matching names and records. In KDD workshop on data cleaning and object consolidation (Vol. 3, pp. 73–78).

de Assis Costa, G., & de Oliveira, J. M. P. (2016). A blocking scheme for entity resolution in the semantic web. In *Advanced Information networking and applications (AINA), 2016 IEEE 30th international conference on* (pp. 1138–1145).

Do, H.-H., & Rahm, E. (2002). COMA: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international conference on Very Large Data Bases* (pp. 610–621). doi:10.1016/B978-155860869-6/50060-3

Dong, X., Halevy, A., & Madhavan, J. (2005). Reference reconciliation in complex information spaces. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (pp. 85–96). doi:10.1145/1066157.1066168

Draisbach, U., & Naumann, F. (2010). DuDe: The duplicate detection toolkit. In *Proceedings of the International Workshop on Quality in Databases (QDB).*

Dunning, T. E., Kindig, B. D., Joshlin, S. C., & Archibald, C. P. (2011). *Associating and linking compact disc metadata*. Google Patents.

Efthymiou, V., Papadakis, G., Papastefanatos, G., Stefanidis, K., & Palpanas, T. (2017). Parallel meta-blocking for scaling entity resolution over big heterogeneous data. *Information Systems*, *65*, 137–157. doi:10.1016/j.is.2016.12.001

Elmagarmid, A., Ilyas, I. F., Ouzzani, M., Quiané-Ruiz, J.-A., Tang, N., & Yin, S. (2014). NADEEF/ER: Generic and interactive entity resolution. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (pp. 1071–1074). doi:10.1145/2588555.2594511

Enríquez, J. G., Domínguez-Mayo, F. J., Escalona, M. J., Ross, M., & Staples, G. (2017). Entity reconciliation in big data sources: A systematic mapping study. *Expert Systems with Applications*, *80*, 14–27. doi:10.1016/j.eswa.2017.03.010

Fellegi, I. P., & Sunter, A. B. (1969). A Theory for Record Linkage. *Journal of the American Statistical Association*, *64*(328), 1183–1210. doi:10.1080/01621459.1969.10501049

Fisher, J., Christen, P., Wang, Q., & Rahm, E. (2015). A clustering-based framework to control block sizes for entity resolution. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 279–288). doi:10.1145/2783258.2783396

Frontini, F., Brando, C., & Ganascia, J.-G. (2015). Domain-adapted named-entity linker using Linked Data. In *Workshop on NLP Applications: Completing the Puzzle co-located with the 20th International Conference on Applications of Natural Language to Information Systems (NLDB 2015).*

Gravano, L., Ipeirotis, P. G., Jagadish, H. V., Koudas, N., Muthukrishnan, S., & Srivastava, D. et al. (2001). *Approximate string joins in a database (almost) for free* (Vol. 1, pp. 491–500). VLDB.

Guha, S., Koudas, N., Marathe, A., & Srivastava, D. (2004). Merging the results of approximate match operations. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30* (pp. 636–647). doi:10.1016/B978-012088469-8.50057-7

Hall, P. A. V., & Dowling, G. R. (1980). Approximate string matching. [CSUR]. *ACM Computing Surveys*, *12*(4), 381–402. doi:10.1145/356827.356830

Harron, K., Goldstein, H., & Dibben, C. (2015). *Methodological developments in data linkage*. John Wiley & Sons. doi:10.1002/9781119072454

Hartnett, J. (2015). Discogs. com. *The Charleston Advisor*, *16*(4), 26–33. doi:10.5260/chara.16.4.26

Hemerly, J. (2011). Making metadata: The case of MusicBrainz.

Hernández, M. A., & Stolfo, S. J. (1995). The merge/purge problem for large databases. *SIGMOD Record*, *24*(2), 127–138. doi:10.1145/568271.223807

Jurczyk, P., Lu, J. J., Xiong, L., Cragan, J. D., & Correa, A. (2008). FRIL: A tool for comparative record linkage. *AMIA ... Annual Symposium Proceedings / AMIA Symposium. AMIA Symposium*, *2008*, 440. PMID:18998844

Kalashnikov, D. V., & Mehrotra, S. (2006). Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Transactions on Database Systems*, *31*(2), 716–767. doi:10.1145/1138394.1138401

Kang, H., Getoor, L., Shneiderman, B., Bilgic, M., & Licamele, L. (2008). Interactive entity resolution in relational data: A visual analytic tool and its evaluation. *Visualization and Computer Graphics. IEEE Transactions on*, *14*(5), 999–1014.

Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., & Jurafsky, D. (2013). Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, *39*(4), 885–916. doi:10.1162/COLI_a_00152

Monge, A. E., Elkan, C., & Associates. (1996). The Field Matching Problem: Algorithms and Applications. In KDD (pp. 267–270).

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys*, *33*(1), 31–88. doi:10.1145/375360.375365

Newcombe, H. B., & Kennedy, J. M. (1962). Record linkage: Making maximum use of the discriminating power of identifying information. *Communications of the ACM*, *5*(11), 563–566. doi:10.1145/368996.369026

Ng, V., & Cardie, C. (2002). Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 104–111).

Nguyen, K., & Ichise, R. (2016). Linked data entity resolution system enhanced by configuration learning algorithm. *IEICE Transactions on Information and Systems*, *99*(6), 1521–1530. doi:10.1587/transinf.2015EDP7392

Papadakis, G., Ioannou, E., Niederée, C., Palpanas, T., & Nejdl, W. (2012). Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data. In *Proceedings of the fifth ACM international conference on Web search and data mining* (pp. 53–62). doi:10.1145/2124295.2124305

Peng, T., Li, L., & Kennedy, J. (2014). A Comparison of Techniques for Name Matching. [JoC]. *Journal on Computing*, *2*(1), 55–61.

Rahmani, H., Ranjbar-Sahraei, B., Weiss, G., & Tuyls, K. (2016). Entity resolution in disjoint graphs: An application on genealogical data. *Intelligent Data Analysis*, *20*(2), 455–475. doi:10.3233/IDA-160814

Rastogi, V., Dalvi, N., & Garofalakis, M. (2011). Large-scale collective entity matching. *Proceedings of the VLDB Endowment*, *4*(4), 208–218. doi:10.14778/1938545.1938546

Sarawagi, S., & Bhamidipaty, A. (2002). Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 269–278). doi:10.1145/775047.775087

Schnell, R., Bachteler, T., & Reiher, J. (2009). Privacy-preserving record linkage using Bloom filters. *BMC Medical Informatics and Decision Making*, *9*(1), 41. doi:10.1186/1472-6947-9-41 PMID:19706187

Shin, K., Jung, J., Lee, S., & Kang, U. (2015). Bear: Block elimination approach for random walk with restart on large graphs. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (pp. 1571–1585). doi:10.1145/2723372.2723716

Song, D., Luo, Y., & Heflin, J. (2017). Linking heterogeneous data in the semantic web using scalable and domain-independent candidate selection. *IEEE Transactions on Knowledge and Data Engineering*, *29*(1), 143–156. doi:10.1109/TKDE.2016.2606399

Stutzbach, A. R. (2011). MusicBrainz [review]. *Notes*, *68*(1), 147–151. doi:10.1353/not.2011.0134

Swartz, A. (2002). Musicbrainz: A semantic web service. *IEEE Intelligent Systems*, *17*(1), 76–77. doi:10.1109/5254.988466

Tejada, S., Knoblock, C. A., & Minton, S. (2001). Learning object identification rules for information integration. *Information Systems*, *26*(8), 607–633. doi:10.1016/S0306-4379(01)00042-4

Tejada, S., Knoblock, C. A., & Minton, S. (2002). 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* (pp. 350–359). doi:10.1145/775047.775099

Vatsalan, D., Christen, P., & Verykios, V. S. (2013). A taxonomy of privacy-preserving record linkage techniques. *Information Systems*, *38*(6), 946–969. doi:10.1016/j.is.2012.11.005

Vatsalan, D., Sehili, Z., Christen, P., & Rahm, E. (2017). Privacy-Preserving Record Linkage for Big Data: Current Approaches and Research Challenges. In Handbook of Big Data Technologies (pp. 851–895). Springer.

Volz, J., Bizer, C., Gaedke, M., & Kobilarov, G. (2009). *Silk-A Link Discovery Framework for the Web of Data* (Vol. 538). LDOW.

Winkler, W. E. (2014). Matching and record linkage. *Wiley Interdisciplinary Reviews: Computational Statistics*, *6*(5), 313–325. doi:10.1002/wics.1317

Yancey, W. E. (2002). BigMatch: A program for extracting probable matches from a large file for record linkage. *Computing*, *1*, 1–8.

Yu, M., Li, G., Deng, D., & Feng, J. (2016). String similarity search and join: A survey. *Frontiers of Computer Science*, *10*(3), 399–417. doi:10.1007/s11704-015-5900-5

Zhu, L., Ghasemi-Gol, M., Szekely, P., Galstyan, A., & Knoblock, C. A. (2016). Unsupervised Entity Resolution on Multi-type Graphs. In *Proceedings of the International Semantic Web Conference* (pp. 649–667).

ENDNOTES

[1] Retrieved 2017-07-21 from https://www.discogs.com/search/advanced