# Do your Resources Sound Similar?

## On the Impact of Using Phonetic Similarity in Link Discovery

Abdullah Fathi Ahmed
Paderborn University, Data Science
Paderborn, Germany
Leipzig University, Computer Science
Leipzig, Germany
afaahmed@mail.upb.de

Mohamed Ahmed Sherif
Paderborn University, Data Science
Paderborn, Germany
Leipzig University, Computer Science
Leipzig, Germany
mohamed.sherif@upb.de

Axel-Cyrille Ngonga Ngomo
Paderborn University, Data Science
Paderborn, Germany
Leipzig University, Computer Science
Leipzig, Germany
axel.ngonga@upb.de

## ABSTRACT

An increasing number of heterogeneous datasets abiding by the Linked Data paradigm is published everyday. Discovering links between these datasets is thus central to achieving the vision behind the Data Web. Declarative Link Discovery (LD) frameworks rely on complex Link Specification (LS) to express the conditions under which two resources should be linked. Complex LS combine similarity measures with thresholds to determine whether a given predicate holds between two resources. State of the art LD frameworks rely mostly on string-based similarity measures such as *Levenshtein* and *Jaccard*. However, string-based similarity measures often fail to catch the similarity of resources with phonetically similar property values when these property values are represented using different string representation (e.g., names and street labels). In this paper, we evaluate the impact of using phonetics-based similarities in the process of LD.

Moreover, we evaluate the impact of phonetic-based similarity measures on a state-of-the-art machine learning approach used to generate LS. Our experiments suggest that the combination of string-based and phonetic-based measures can improve the F-measures achieved by LD frameworks on most datasets.

## CCS CONCEPTS

• **Information systems** → *Similarity measures.*

## KEYWORDS

Link Discovery, Link Specification,Similarity Measure, Phonetic Algorithms, Machine Learning

**ACM Reference Format:**
Abdullah Fathi Ahmed, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. 2019. Do your Resources Sound Similar? : On the Impact of Using Phonetic Similarity in Link Discovery. In *10th International Conference on Knowledge Capture (K-CAP'19), November 19–21, 2019, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3360901.3364426


*K-CAP '19, November 19–21, 2019, Marina Del Rey, CA, USA*

ACM ISBN 978-1-4503-7008-0/19/11…$15.00
https://doi.org/10.1145/3360901.3364426

## 1 INTRODUCTION

With the rapid increase in the number and heterogeneity of RDF datasets comes the need to link such datasets. Declarative LD frameworks depend on complex link specification (LS) to express the conditions necessary for linking resources within these datasets. For instance, state-of-the-art LD frameworks such as Limes [23] and Silk [15] adopt a property-based computation of links between entities. To make sure that links can be computed with high accuracy, these frameworks provide a large number of similarity measures (e.g., Levenshtein, Jaccard for strings) for comparing property values. One of the most common tasks of LD frameworks is to discover the same resources in the same or different dataset(s). Such resources are commonly linked via `owl:sameAs`. The discovery of such links is often dubbed *matching*. The input for a matching task consists of a source knowledge base $S$, a target knowledge base $T$, a similarity measure[1] $\sigma$ and a threshold $\theta$. Consequently, a key challenge arises when trying to discover matches between resources in $S$ and in $T$, which is the selection of appropriate similarity measures. While the selection process could be carried out manually, finding the appropriate $\sigma$ and $\theta$ is often a tedious endeavor. This is due amongst other to the wide range of measures available in the literature (e.g., [12] categorize similarity measures into different groups based on their characteristics while [17, 39] survey string similarity search and joins).

Supporting the user during the process of finding appropriate similarity measures and thresholds for matching tasks has been studied in a large body of literature. Recently, many LD frameworks have adopted machine learning approaches to support their user in their search for adequate similarity measures via supervised, unsupervised and even active learning techniques [14, 16, 25–28, 35, 38]. While phonetic similarity measures have been applied to different challenges in information retrieval (e.g., the representation of Short Message Services (SMS) [3, 32], *microtext* normalization in social networks on the Web [10]), their use in LD (and especially matching) has not been studied in the current literature.

In this paper, we study the impact of phonetic similarities on the LD problem. In particular, we aim to answer the following research question: *Can the inclusion of phonetic similarities in LS improve the F-measure of LD frameworks?* To address this question, we add phonetic similarities into a LD framework and use them to measure the similarity between written resources, hence measuring whether phonetic information can actually improve the results of LD. Throughout our experiments, we limit ourselves to studying

[1] Note that, a distance measure can be used instead of a similarity measure.

the effect of including phonetic-based similarities into LSs used by declarative LD frameworks to express the conditions necessary for matching resources from heterogeneous datasets. To the best of our knowledge, this is the first work that actually quantifies the effect of phonetic algorithms on LD.

The contributions of this paper can be summarized as follows:

- We extend a state-of-the-art LD framework with phonetic similarities and measure the effect of this addition on the overall F-measure achieved by said framework.
- To measure the effect of phonetic similarities, we evaluate the F-measure achieve by the same state-of-the-art algorithm in two configurations: (1) using only string similarities and (2) using string and phonetic similarities.

The rest of this paper is structured as follows: First, we introduce the preliminaries and background to this paper in Section 2. Then, we give an overview of the phonetic similarities used in this paper in Section 3. We then evaluate and discuss the impact of phonetic similarity with respect to the *F-Measure* on benchmark datasets in Section 4. After a brief review of related work in Section 5, we conclude our work with some final remarks and future work in Section 6.

## 2 PRELIMINARIES

In the following, we present the core of the formalization and notation necessary to achieve the goal of our work. We first define LD. Then, we present the syntax and the semantics of the grammar that underlies LS.

### 2.1 Link Discovery

Let $K$ be a finite RDF *knowledge base*. $K$ is as a set of triples $(s, p, o) \in (\mathcal{R} \cup \mathcal{B}) \times \mathcal{P} \times (\mathcal{R} \cup \mathcal{L} \cup \mathcal{B})$, where $\mathcal{R}$ is the set of all resources, $\mathcal{B}$ is the set of all blank nodes, $\mathcal{P}$ is the set of all predicates and $\mathcal{L}$ the set of all literals. The LD problem can be expressed as follows: Given two sets of resources $S$ and $T$ (e.g., author and writer) and a relation $r$ (e.g., `owl:sameAs`), find all pairs $(s, t) \in S \times T$ such that $r(s, t)$ holds. The result is produced as a set of links called a *mapping*: $M_{S,T} = \{(s_i, r, t_j) | s_i \in S, t_j \in T\}$. Optionally, a similarity score ($sim \in [0, 1]$) calculated by an LD framework can be added to the entries of mappings to express the framework's confidence in a computed link.

### 2.2 Link Specification

The goal of LD is to find the set $\{(s, t) \in S \times T : r(s, t)\}$. Declarative LD frameworks define the conditions necessary to generate such links using LS. Several grammars have been used for describing LS in previous work [25, 35]. These grammars suppose that a LS consists of two types of atomic components: *similarity measures* $m$, which compare property values of input resources and *operators* $op$, which can be used to combine these similarities to more complex specifications. We define a similarity measure $m$ as a function $m : S \times T \rightarrow [0, 1]$. We use *mappings* $M_{S,T} \subseteq S \times T$ to collect the results of the application of a similarity function to $S \times T$ or subsets thereof. We define a *filter* as a function $f(m, \theta)$. We call a LS *atomic* iff it contains exactly one filtering function. A *complex LS* can be obtained by combining two specifications $L_1$ and $L_2$ through an *operator* $op$ that allows the results of $L_1$ and $L_2$ to be fused. We use the operators $\sqcap$, $\sqcup$ and $\setminus$ as they are complete and frequently used to define LS [35]. A graphical representation of a complex LS is given in Figure 1.

Our definition of the semantics $[[L]]_M$ of a LS $L$ w.r.t. a mapping $M$ is given in Table 1. These semantics are similar to those used in languages like SPARQL, i.e., they are defined extensionally through the mappings they generate. The mapping $[[L]]$ of a LS $L$ with respect to $S \times T$ contains the links that will be generated by $L$.

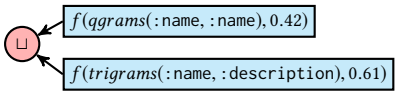


**Figure 1: Example of complex LS. The filter nodes are rectangles while the operator nodes are circles.**

**Table 1: Link Specification Syntax and Semantics.**

| LS | $[[LS]]_M$ |
|---|---|
| $f(m, \theta)$ | $\{(s, t) \mid (s, t) \in M \wedge m(s, t) \geq \theta\}$ |
| $L_1 \sqcap L_2$ | $\{(s, t) \mid (s, t) \in [[L_1]]_M \wedge (s, t) \in [[L_2]]_M\}$ |
| $L_1 \sqcup L_2$ | $\{(s, t) \mid (s, t) \in [[L_1]]_M \vee (s, t) \in [[L_2]]_M\}$ |
| $L_1 \setminus L_2$ | $\{(s, t) \mid (s, t) \in [[L_1]]_M \wedge (s, t) \notin [[L_2]]_M\}$ |

## 3 APPROACH

We introduced all ingredients necessary to extend LD frameworks by adopting phonetic algorithms and to assess the effect of such phonetic algorithms on the machine learning approaches used to generate LS. Our work in this paper relies on: First, the phonetic algorithms as a major part to extend our LS and LD framework. Second, machine learning algorithms to generate either a hybrid LS including both string and phonetic similarity measures or only including phonetic algorithms.

### 3.1 Phonetic Algorithms

A phonetic algorithm transforms an input word into a phonetic code which abstracts upon the pronunciation of said word in a particular language. Special attention must be drawn to the approximate nature of these codes, as their goal is to support the matching of words with similar pronunciations. In this work, we utilize the ten most popular state–of–the–art phonetic algorithms developed for the English and German languages. Most of these algorithms were originally designed with the end goal of matching person names [13, 29] and were used as such in previous works [7, 11, 36]. In particular, [7, 11, 36] carry out purely comparative studies of name–matching algorithms while considering phonetic algorithms and other types of similarity metrics. Given that the phonetic phenomena captured by phonetic algorithms occur in most words of the language they were designed for, we hypothesize that phonetic similarities can also be useful in matching other types of similarly sounding words and improving the results of LD.

In this work, we evaluate *Soundex, Metaphone, Double Metaphone, Daitch-Mokotoff soundex, New York State identification and intelligence system (NYSIIS), Match rating approach, Caverphone,*

*Caverphone2, Cologne,* and *Refined soundex* to improve the LD process. In the rest of this section, we explain the algorithms behind the *Soundex, Metaphone, Cologne* and *Daitch-Mokotoff Soundex* phonetic similarities, the rest of the algorithms can be found in [10].

#### 3.1.1 Soundex.

Soundex was developed by Robert C. Russell and Margaret King Odell and patented in 1918 [20]. It is considered the first phonetic algorithm in history. The algorithm mainly encodes the consonants of an input word using numerical digits, but also encodes both consonants and vowels in the first position using that same character. The encoded word consists of a letter followed by three numerical digits: the letter is the first letter of the word, and the digits encode the remaining consonants. Consonants at a similar place of articulation share the same digit. For example, the labial consonants `B, F, P,` and `V` are each encoded as the number 1. The generated codes have a fixed length of four characters, attached with trailing 0's when needed [34]. Soundex is the basis for many other modern phonetic algorithms. These newer algorithms essentially try to address its low precision. A commonly used example is the Refined Soundex algorithm,[2] which is also tested in this work. This revised version does not impose a length limit on the encoding and takes vowels more into consideration for the encoding.

#### 3.1.2 The Cologne phonetic.

This phonetic algorithm, also known as "die Kölner Phonetik", was published in 1969 by Hans Joachim Postel [33]. It is inspired by soundex phonetic algorithm but is optimized to match words written in *German*. One of the core use case which triggered its design was the need to perform similarity search between words. For example, it made finding entries like `"Meier"` under different spellings such as `"Maier", "Mayer"`, or `"Mayr"` in name lists.

Like Soundex, the algorithm assigns a sequence of digits (phonetic code) to each word, so that identically sounding words have the same code. As shown in Table 2, each letter of a word matches with a digit between "0" and "8". To select the appropriate digit, we need to use at most one adjacent letter as a context. Furthermore, a set of rules must be applied to the first letter of words hence the similar sounds are supposed to be assigned the same code. For instance, the letters `"W"` and `"V"` are both encoded with the number "3" (e.g. the phonetic code for `"Wikipedia"` is "3412" ($W = 3, K = 4, P = 1$, and $D = 2$)).

#### 3.1.3 Metaphone.

Metaphone considers sets of letters to identify the phonetic variations and inconsistencies in words [30]. The algorithm initially performs transformations using diphthongs such as converting `"MB"` into `"B"` if at the end of the word, `"SCH"` into `"K", "CIA"` into `"X"`, and removes all the vowels in the encoded word. The algorithm shows that the phonetic sound of vowels combined with the consonants is considered instead of individual consonant or vowel sounds. Metaphone imposes a length code from 4-letter code to 12-letter code [4]. The metaphone code used in this paper is of 4-letter code length. In 2000, a new revised version of *Metaphone*, dubbed *Double Metaphone* [31], was released with improvements

[2]Made available by A.S. Foundation since 2017 as an Apache commons codec. http://commons.apache.org/proper/commons-codec/ (accessed February 2019).

**Table 2: Cologne phonetic encoding**

| Letter | Context | Code |
|---|---|---|
| A, E, I, J, O, U, Y | | 0 |
| H | | - |
| B | | 1 |
| P | Not before H | 1 |
| D, T | Not before C, S, Z | 2 |
| F, V, W | | 3 |
| P | Before H | 3 |
| G, K, Q | | 4 |
| C | In the initial sound before A, H, K, L, O, Q, R, U, X | 4 |
| C | Before A, H, K, O, Q, U, X except after S, Z | 4 |
| X | Not after C, K, Q | 48 |
| L | | 5 |
| M, N | | 6 |
| R | | 7 |
| S, Z | | 8 |
| C | After S, Z | 8 |
| C | In initial position except before A, H, K, L, O, Q, R, U, X | 8 |
| C | Not before A, H, K, O, Q, U, X | 8 |
| D, T | Before C, S, Z | 8 |
| X | After C, K, Q | 8 |

such as the consideration of several spelling peculiarities from different languages, including English. *Double Metaphone* can provide an output of two alternative encodings of the input word.

#### 3.1.4 Daitch-Mokotoff Soundex.

The Daitch-Mokotoff Soundex is designed to improve the original *Soundex* algorithm by increasing its precision when dealing with *Slavic* and *Yiddish* surnames [21]. In contrast to the original *Soundex*, it assigns code with lengths up to six characters. The algorithm codes the initial character of the name and applies several rules to encode multiple character n-grams as single units. For instance, the code of `Moskowitz` in the original Soundex is `M232`, while it is computed to be 645740 by Daitch-Mokotoff Soundex. This code is the same as the code of the word `Moskovitz`, which has a different *Soundex* code (i.e., `M213`).

### 3.2 Machine Learning for LD

Our machine learning evaluation relies on Wombat [35]. Wombat is based on generalization via an upward refinement operator to traverse the space of LS. Wombat consists of two main steps. First, it aims to derive initial atomic specifications $A_i$ that achieve a target function (e.g., the F-measure when provided with training data or a pseudo-F-measure when used in an unsupervised setting). The second step of the approach combines these atomic specifications to a complex LS by using the operators $\sqcap$, $\sqcup$ and $\setminus$. For more details see [35]. We chose this algorithm because it achieves state-of-the-art performance while being deterministic. Moreover, it is the only positive-only learning algorithm for LD found in the current literature.

## 4 EVALUATION

To evaluate the impact of phonetic-based similarities on the effectiveness of LD, we conducted a set of experiments to answer the following two research questions:

$Q_1$: Is using phonetic algorithms in LD useful, i.e., can we achieve a higher *F-Measure* using LSs that combine phonetic-based similarities with string similarities than when using LSs which rely exclusively on string similarities?

$Q_2$: Do phonetic-based similarities enhance the performance of machine learning algorithms for generating LSs?

## 4.1 Experimental setup

To answer our research questions, we carried out four experiments using the state-of-the-art LD framework Limes.[3] We configured Wombat as follows: We set the maximal depth of refinement to 2, minimum property coverage to 0.4, maximum iteration time 20 minutes, and the maximum refinement trees size to 1000. Our implementation of phonetic-based similarities are based on the *Apache commons codec package*.[4] We ran all experiments using a 3.40 GHz Intel Core i5 processor (4 cores) with 8 GB of memory. We used the *F-Measure* as our evaluation metric. Our experiments were carried out on benchmark datasets from [19], i.e., `Amazon-GoogleProducts`, `DBLP-ACM` and `ABT-BUY` as well as on the OAEI datasets `Person1`, `Person2 and Restaurants` and on the `Drugs` dataset. [5] We used these benchmark datasets because they are widely used both in the Link Discovery and the entity matching literature (see [19, 22] and are available as RDF. The datasets from [19] were scrapped from the Web and curated manually (see refered paper for details). Hence, they are representative for real-world data and allow measuring the impact of phonetic similarity algorithms in real-world scenarios. The other datasets are synthetic but were also not designed to suit particular kinds of algorithms. While datasets designed to evaluate phonetic algorithms do exist (see, e.g., [10]), they were not considered in our experiments because they are clearly biased towards phonetic algorithms. Hence, the results we would achieve on such datasets would not reflect the real performance of LD frameworks enriched with phonetic algorithms on real datasets.

*Experiment 1* was conducted to answer the $Q_1$ while *Experiment 2, 3* and *4*, were designed to answer $Q_2$. In the following, we explain each experiment in detail.

- **Experiment 1: Atomic LS.** This experiment consists of 16 tasks. In each task, we consider one atomic similarity measure (i.e., either a string similarity or a phonetic similarity measure). For phonetic similarity measures, we evaluated the 10 most widely used phonetic algorithms (i.e., *Soundex, Metaphone, Double Metaphone, Daitch-Mokotoff soundex, New York State Identification and Intelligence System (NYSIIS), Match rating approach, Caverphone, Caverphone2, Cologne* and *Refined soundex*). For evaluating string similarity measures, we picked up the 6 most used string similarities in the recent literature of LD [25, 35]. i.e., *cosine, Overlap, Trigram, Qgram, Levenstein* and *Jaccard*.
- **Experiment 2: Complex string-based LS.** In this experiment, we used the unsupervised version of the simple Wombat operator [35] to generate a set of LS to be used later in the process of linking datasets. Each generated LS relies on a combination of the string similarity measures *Jaccard, Trigrams, Cosine,* and *Qgrams*. We kept the same initial setting as defined in Wombat [35]. Moreover, we limit the set of properties to be included in the learning process of Wombat to include only non-numerical properties. i.e., {`name`, `authors`, `description`, `surname`, `title`, `given-name`}.
- **Experiment 3: Complex phonetic-based LS.** We repeated Experiment 2 but provided Wombat exclusively with the 10 phonetic measures aforementioned. Consequently, the LS learned contained one or more phonetic similarity measures. We used exactly the same set of properties as listed in *Experiment 2*.
- **Experiment 4: Complex hybrid LS.** In this last experiment, we also used the unsupervised simple Wombat approach but allowed Wombat to compute LSs using both string and phonetic-based similarity measures. We dubbed the generated LS *hybrid*. For this experiment, we used all the string similarity measures mentioned in *Experiments 2* and all the phonetic algorithms in *Experiment 3*. We used the same set of properties as in *Experiment 1*.

## 4.2 Results and Discussion

***Answering*** $Q_1$**:** We start our discussion by analyzing the results from *Experiment 1* to answer $Q_1$ as follows:

Table 3 shows the results obtained from *Experiment 1* where the upper part of the table (colored in dark gray) provides the results of the phonetic measures, while the results of the string similarity measures are in the lower part of the table (colored in light gray). The results clearly show that for the dataset `Drugs`, all the phonetic algorithms achieved a higher *F-Measure* than string similarity measures. The achieved *F-Measure* is up to 50% higher. This result is due to a peculiarity of the `Drugs` dataset: Here, the one property (i.e., `name`) is sufficient to characterize a drug. Given that the names of drugs are short, the phonetic encoding of a name is sufficient to represent a significant portion of the name. Hence, matching these names phonetically is easy. Another remarkable improvement over string measures is observed for the titles in DBLP-ACM (see `DBLP-ACM (Title)`) in which algorithms such as *Caverphone2* and *Match rating approach (Match)* achieved up to 40% higher *F-Measure*, while for algorithms such as *Refined soundex (Rsoundex)* and *Cologne*, the improvement of *F-Measure* reached up to 50% for the same dataset. For *Caverphone2*, the length of encoded string is a fixed length of 10 characters, while the length code in *Match rating approach (Match)* is fixed to a length of 6. On the other hand, the algorithms *Refined soundex (Rsoundex)* and *Cologne* impose no limit on code length, however we fixed the length to 10 characters. Not surprisingly, the length of the encoded generated by a phonetic similarity can impact the *F-Measure*, especially in cases where the ratio between the length of encoded string and the length of string is large.

The results obtained in Experiment 1 suggest that using phonetic algorithms in LD has the potential of enhancing the effectiveness of LD frameworks. It is important to mention that generated codes by all the phonetic algorithms used have a fixed length and in most cases they have length between 4 and 10 digits. The length of generated codes can notably affect the effectiveness of linking knowledge graphs especially when the length of strings (e.g., name of persons or cities) to be encoded phonetically is long. We conducted more experiments to test different values of $\theta$. The results with $\theta = 0.9$

[3] https://github.com/dice-group/LIMES (version 1.5.5, accessed February 2019)
[4] http://commons.apache.org/proper/commons-codec/, (accessed February 2019).
[5] http://www4.wiwiss.fu-berlin.de/drugbank/sparql

are shown in Table 4. Table 5 shows the result when $\theta = 0.6$.[6]. We also computed *Recall* and *Precision*.[7]

***Answering $Q_2$:*** To answer $Q_2$, we used the results obtained from *Experiments 2,3 & 4*. In Table 6, we present the LSs computed using only string-based measures relying on the simple version of Wombat algorithm. Table 7 shows the LSs which resulted from using phonetic-based measures exclusively. The results in Table 6 and 7 show different behavior for the same datasets during LD process leading to achieving different *F-Measure*. It is worth noting that phonetic approaches which generate short encodings tend to restrict the effectiveness of Wombat. This is clearly due to the discrete value space to which these similarities map pairs of input strings. For example, if the length of the code is 4 (like for Soundex), there are exactly 5 similarity values of similarity that two strings can be mapped to (i.e., 0, 0.25, 0.5, 0.75 and 1.0). While this accelerates the convergence of Wombat, the very low granularity of the similarity sores also reduces the number of LSs with different output, and hence also the probability of achieving a high *F-Measure*. Overall, our results suggest that using phonetic similarities alone in LD is rarely useful, as they tend to perform poorly on complex datasets (e.g., Amazon-GoogleProducts).

We hence also studied the combination of string-based and phonetic measures. Our results are presented in Table 8, which shows the resulting LSs computed by Wombat. The results in Table 8 indicate that using hybrid LS has a measurable (not always positive) impact on the LSs computed by Wombat.

As shown in Figure 2, Wombat achieve a higher F-measure on most of the 4 real datasets when using hybrid similarity measures. For instance, the dataset `Person1` achieves up to 11% higher *F-Measure* over the *F-Measure* obtained from only using phonetic-based LS. For the datasets `ABT-BUY` and `Restaurants`, the string-based approach achieves a higher *F-Measure*.

We believe that the lower *F-Measure* for datasets `ABT-BUY` and `Restaurants` in the case of using a hybrid-based approach is due to: (1) the limited length of the encoding generated by some of the phonetic algorithms, which limits the flexibility of Wombat algorithm and (2) the significantly larger search space of Wombat (because of the combination of 4 string similarity measures and 10 phonetic algorithms). (2) is the most relevant here: Because Wombat is configured to only search a limited space, it ends up not swapping out the phonetic similarity for a string similarity. However, note that Wombat (especially the complete version of the operator) is guaranteed to find any specification which can be built from the atomic specifications it is provided. However, this search does not always scale. We hence aimed to report the results of the approach with a realistic configuration for Wombat. In our larger-scale experiments (which will roughly 6 months to run), we will allow Wombat to explore significantly larger portions of the search space in a supervised setting, run a ten-fold cross-validation and characterize when the approach discards phonetic similarities

These two points clearly show the limitations brought about by using phonetic similarities in an unsupervised setting as in our experiments. We will address these limitations in future works by devising a customized machine learning algorithm that can handle phonetic algorithms with different lengths of encoded words. Second, we will study other exploration strategies that allow relaxing the termination condition of the algorithm, such as the depth of generated LS.

[6]More results with varying values of $\theta$ can be accessed at https://bit.ly/2DUWTUL.
[7]All results where *Recall* and *Precision* are reported at https://bit.ly/2DUWTUL.

**Table 3: *F-Measure* results achieved when applying string and phonetic similarity measures with $\theta = 1.0$ (perfect matches). The F-measure achieved by a phonetic similarity being in bold signifies that this F-measure is superior to that achieved by string-based measure. The same holds vice-versa.**

| Algorithm | DBLP-GoogleScholar (Title) | DBLP-GoogleScholar (Authors) | Amazon-GoogleProducts (Products) | Drugs | DBLP-ACM (Title) | DBLP--ACM (Authors) | Person1 (Surname) | Person2 (Surname) | ABT-BUY (Name) | ABT-BUY (Description) |
|---|---|---|---|---|---|---|---|---|---|---|
| Caverphone1 | 0.16 | 0.17 | 0.06 | **0.97** | 0.52 | 0.24 | 0.6 | 0.42 | 0.14 | 0.06 |
| **Caverphone2** | **0.65** | 0.23 | **0.21** | **0.98** | **0.88** | 0.34 | 0.67 | 0.45 | 0.16 | 0.04 |
| Metaphone | 0.04 | 0.15 | 0.05 | **0.83** | 0.22 | 0.14 | 0.68 | **0.49** | 0.07 | 0.05 |
| Double Metaphone | 0.04 | 0.12 | 0.05 | **0.78** | 0.21 | 0.13 | 0.63 | 0.47 | 0.06 | 0.04 |
| Soundex | 0.03 | 0.06 | 0.04 | **0.72** | 0.2 | 0.1 | 0.64 | **0.54** | 0.05 | 0.04 |
| **Rsoundex** | **0.59** | **0.38** | 0.12 | **0.98** | **0.9** | 0.31 | 0.71 | 0.43 | 0.06 | 0.02 |
| **Cologne** | **0.59** | **0.4** | 0.12 | **0.96** | **0.9** | 0.33 | 0.63 | 0.38 | 0.06 | 0.02 |
| Nysiis | 0.05 | 0.22 | 0.05 | **0.88** | 0.25 | 0.15 | 0.72 | 0.47 | 0.07 | 0.05 |
| Daitch | 0.13 | 0.25 | 0.06 | **0.96** | 0.47 | 0.23 | 0.65 | 0.47 | 0.14 | 0.07 |
| **Match** | 0.46 | **0.41** | 0.12 | **0.98** | **0.81** | 0.35 | 0.74 | 0.45 | **0.36** | 0 |
| Cosine | 0.53 | 0.32 | 0.07 | 0.48 | 0.57 | **0.61** | **0.81** | 0.48 | 0.01 | 0 |
| Overlap | 0.53 | 0.32 | 0.07 | 0.48 | 0.57 | **0.61** | **0.81** | 0.48 | 0.01 | 0 |
| Trigram | 0.53 | 0.32 | 0.07 | 0.48 | 0.57 | **0.61** | **0.81** | 0.48 | 0.01 | 0 |
| Jaccard | 0.53 | 0.32 | 0.07 | 0.48 | 0.57 | **0.61** | **0.81** | 0.48 | 0.01 | 0 |
| Levenshtein | 0.5 | 0.31 | 0.06 | 0.48 | 0.56 | 0.34 | **0.81** | 0.48 | 0.01 | 0 |
| Qgram | 0.51 | 0.31 | 0.06 | 0.48 | 0.56 | 0.35 | **0.75** | 0.31 | 0.01 | 0 |

**Table 4: *F-Measure* results achieved using string and phonetic similarity measures with $\theta = 0.9$.**

| Algorithm | DBLP-GoogleScholar (Title) | DBLP-GoogleScholar (Authors) | Amazon-GoogleProducts (Products) | Drugs | DBLP-ACM (Title) | DBLP-ACM (Authors) | Person1 (Surname) | Person2 (Surname) | ABT-BUY (Name) | ABT-BUY (Description) |
|---|---|---|---|---|---|---|---|---|---|---|
| Caverphone1 | 0.09 | 0.10 | 0.03 | 0.95 | 0.36 | 0.18 | 0.46 | 0.40 | 0.09 | 0.08 |
| Caverphone2 | 0.55 | 0.16 | 0.21 | 0.97 | 0.80 | 0.34 | 0.54 | 0.46 | 0.38 | 0.53 |
| Metaphone | 0.02 | 0.09 | 0.03 | 0.72 | 0.13 | 0.08 | 0.70 | 0.61 | 0.03 | 0.03 |
| DoublMetaphone | 0.02 | 0.06 | 0.03 | 0.64 | 0.12 | 0.08 | 0.64 | 0.53 | 0.03 | 0.03 |
| Soundex | 0.02 | 0.03 | 0.02 | 0.56 | 0.11 | 0.06 | 0.50 | 0.55 | 0.03 | 0.02 |
| RSoundex | 0.75 | 0.68 | 0.47 | 0.99 | 0.87 | 0.46 | 0.76 | 0.57 | 0.88 | 0.90 |
| Cologne | 0.75 | 0.69 | 0.46 | 0.97 | 0.87 | 0.57 | 0.68 | 0.52 | 0.85 | 0.75 |
| Nysiis | 0.03 | 0.14 | 0.03 | 0.79 | 0.14 | 0.09 | 0.76 | 0.66 | 0.03 | 0.03 |
| Daitch | 0.07 | 0.16 | 0.03 | 0.93 | 0.31 | 0.16 | 0.51 | 0.47 | 0.09 | 0.08 |
| Match | 0.36 | 0.45 | 0.20 | 0.98 | 0.70 | 0.37 | 0.74 | 0.59 | 0.44 | 0.33 |
| Cosine | 0.83 | 0.40 | 0.58 | 1.00 | 0.92 | 0.59 | 0.78 | 0.79 | 1.00 | 0.00 |
| Overlap | 0.83 | 0.40 | 0.58 | 1.00 | 0.92 | 0.59 | 0.78 | 0.79 | 1.00 | 0.00 |
| Trigram | 0.83 | 0.40 | 0.58 | 1.00 | 0.92 | 0.59 | 0.78 | 0.79 | 1.00 | 0.00 |
| Jaccard | 0.82 | 0.39 | 0.44 | 1.00 | 0.92 | 0.57 | 0.78 | 0.79 | 1.00 | 0.00 |
| Levenshtein | 0.82 | 0.40 | 0.43 | 1.00 | 0.92 | 0.44 | 0.78 | 0.79 | 1.00 | 0.00 |
| Qgram | 0.83 | 0.40 | 0.44 | 1.00 | 0.91 | 0.54 | 0.67 | 0.26 | 1.00 | 0.00 |

Table 5: *F-Measure* results of applying string and phonetic similarity measures with $\theta$ = 0.6.

| Algorithm | DBLP-GoogleScholar (Title) | DBLP-GoogleScholar (Authors) | Amazon-GoogleProducts (Products) | Drugs | DBLP-ACM (Title) | DBLP-ACM (Authors) | Person1 (Surname) | Person2 (Surname) | ABT-BUY (Name) | ABT-BUY (Description) |
|---|---|---|---|---|---|---|---|---|---|---|
| Caverphone1 | 0.00 | 0.00 | 0.01 | 0.11 | 0.05 | 0.04 | 0.01 | 0.01 | 0.02 | 0.02 |
| Caverphone2 | 0.19 | 0.00 | 0.09 | 0.15 | 0.55 | 0.20 | 0.02 | 0.01 | 0.14 | 0.14 |
| Metaphone | 0.00 | 0.01 | 0.01 | 0.13 | 0.04 | 0.02 | 0.13 | 0.10 | 0.02 | 0.02 |
| DoublMetaphone | 0.00 | 0.00 | 0.01 | 0.10 | 0.03 | 0.02 | 0.10 | 0.08 | 0.02 | 0.02 |
| Soundex | 0.00 | 0.00 | 0.01 | 0.07 | 0.02 | 0.01 | 0.08 | 0.08 | 0.01 | 0.01 |
| RSoundex | 0.67 | 0.23 | 0.36 | 0.80 | 0.82 | 0.40 | 0.30 | 0.18 | 0.80 | 0.03 |
| Cologne | 0.04 | 0.00 | 0.02 | 0.01 | 0.18 | 0.01 | 0.01 | 0.00 | 0.17 | 0.00 |
| Nysiis | 0.00 | 0.00 | 0.01 | 0.03 | 0.02 | 0.01 | 0.02 | 0.02 | 0.01 | 0.01 |
| Daitch | 0.02 | 0.02 | 0.02 | 0.42 | 0.12 | 0.08 | 0.09 | 0.08 | 0.03 | 0.03 |
| Match | 0.01 | 0.01 | 0.03 | 0.10 | 0.05 | 0.05 | 0.02 | 0.02 | 0.04 | 0.00 |
| Cosine | 0.58 | 0.20 | 0.40 | 0.97 | 0.81 | 0.30 | 0.78 | 0.79 | 0.40 | 0.00 |
| Overlap | 0.59 | 0.20 | 0.39 | 0.97 | 0.81 | 0.33 | 0.78 | 0.79 | 0.41 | 0.00 |
| Trigram | 0.59 | 0.20 | 0.39 | 0.97 | 0.81 | 0.33 | 0.78 | 0.79 | 0.41 | 0.00 |
| Jaccard | 0.83 | 0.42 | 0.53 | 1.00 | 0.91 | 0.58 | 0.78 | 0.79 | 0.92 | 0.00 |
| Levenshtein | 0.82 | 0.40 | 0.43 | 1.00 | 0.92 | 0.44 | 0.78 | 0.79 | 1.00 | 0.00 |
| Qgram | 0.76 | 0.34 | 0.55 | 0.91 | 0.85 | 0.52 | 0.63 | 0.27 | 0.68 | 0.00 |

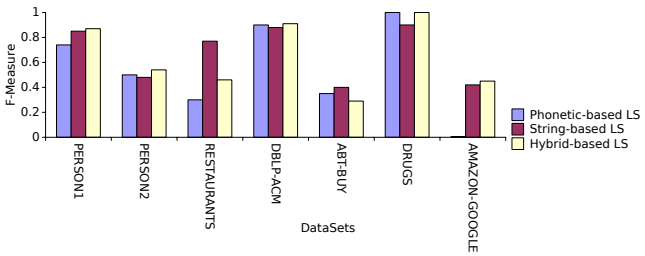


**Figure 2: Phonetic-based LS vs string-based LS vs hybrid LS with respect to the *F-Measure*.**

# 5 RELATED WORK

We believe that this is the first work that aims to quantify the effect of phonetic algorithms on LD and to generate hybrid LS using a machine learning approach. Related work comes from two research areas: Declarative LD and phonetic similarities.

## 5.1 Declarative Link Discovery

LD frameworks rely on complex LS to express the conditions necessary for linking resources within RDF datasets. In the state-of-the-art LD frameworks such as Limes [23] and Silk [15], a property-based computation of links between entities has been adopted. Such frameworks help their users to manually write LS and execute it against source-target resources. In recent years, the problem of using machine learning for the automatic generation of accurate LS has been addressed by most of the LD frameworks. For example, the Silk framework [15] implements a batch learning approach for the discovery LS, based on genetic programming, which is similar to the approach presented in [6]. In Limes framework, the active learning Raven algorithm [24] is implemented to treat the discovery of specifications as a classification problem. In Raven, the discovery of LS is done by first finding class and property mappings between knowledge bases automatically. It then uses these mappings to calculate linear and boolean classifiers that can be used as LS. A related approach that has the goal of detecting the discriminative properties for linking is presented by [37]. In addition, several machine-learning approaches have been developed to learn LS as classifiers for record linkage. For example, machine-learning frameworks such as FEBRL [8] and MARLIN [5] rely on models such as Support Vector Machines [9, 18], decision trees [40] and rule mining [1] to detect classifiers for record linkage.

In most LD approaches for *learning LS*, supervised machine learning methods were developed. One of the first approaches to target this goal was presented in [14]. Although this approach achieves a high *F-Measure*, it also needs huge amounts of training data. Whereas active learning such as Eagle [25], which is based on genetic programming, generates highly accurate LSs while reducing the annotation burden for the user. Hence, methods based on active learning have also been developed (see, e.g., [16, 27]). In general, these approaches assume some knowledge about the type of links to be discovered. For example, unsupervised approaches such as Paris [38] target the discovery of the `owl:sameAs` links exclusively. Modern unsupervised approaches to learn LS propose techniques based on probabilistic models [38] and genetic programming [26, 28], which all suppose that a 1-to-1 mapping is to be discovered. Recently, the Wombat algorithm [35] has implemented a machine leaning algorithm for automatic LS finding by using generalization via an upward refinement operator.

## 5.2 Phonetic Similarities

Phonetic similarities have been traditionally used for phonetic string matching in information retrieval. For instance, the work [41] explains the parallels between information retrieval and phonetic matching. The authors compare different codes and propose new techniques for text representation to be used on the information retrieval task. Moreover, phonetic algorithms play a central role in mobile applications. For instance, the works [3, 32]) use phonetic-based representations for SMS message. In recent work such as [10], phonetic algorithms have been considered to tackle the problem of *microtext* normalization.

# 6 CONCLUSIONS AND FUTURE WORK

In this paper, we present first results on the impact of using phonetic-based similarity measures in LD. Our evaluation suggest that the combination of phonetic similarities and string similarites has the potential to improve the performance of LD approaches, especially on real datasets. In particular, our results suggest that simple LS based on phonetic similarity measures can achieve an up to 50% higher *F-Measure*. We consider these results as a promising milestone toward effective LD tools based on phonetic similarity measures. However, achieving conclusive results allowing to characterize datasets upon which phonetic similarities should be used will demand a significantly larger number of experiments to be carried out. This research question will be addressed in future works.

Table 6: String-based Link-Specification (LS) generated by Wombat algorithm with respective *F-Measure* (F).

| Datasets | F | LS |
|---|---|---|
| Person1 | 0.850 | AND(AND(jaccard(x.surname,y.surname)\|1.0, cosine(x.surname,y.surname)\|1.0)\|0.0,qgrams(x.given-name,y.given-name)\|1.0) |
| Person2 | 0.480 | OR(AND(jaccard(x.surname,y.surname)\|1.0, cosine(x.surname,y.surname)\|1.0)\|0.0,jaccard(x.given-name,y.given-name)\|1.0) |
| Restaurants | 0.770 | OR(AND(jaccard(x.name,y.name)\|0.48,cosine (x.name,y.name)\|0.53)\|0.0,cosine(x.name,y.name)\|0.53) |
| DBLP–ACM | 0.880 | MINUS(qgrams(x.title,y.title)\|0.43,cosine (x.authors,y.title)\|0.48) |
| ABT–BUY | 0.400 | OR(AND(jaccard(x.name,y.name)\|0.43,cosine (x.name,y.name)\|0.6)\|0.0,cosine(x.name,y.name)\|0.6) |
| Drugs | 0.900 | OR(qgrams(x.name,y.name)\|0.66,jaccard(x.name,y.name)\|0.48) |
| Amazon–GoogleProducts | 0.420 | OR(cosine(x.title,y.name)\|0.53,cosine(x.description,y.description)\|0.43) |

Table 7: Phonetic-based LS generated using Wombat algorithm with respect to *F-Measure* (F) and Link-Specification (LS).

| Datasets | F | LS |
|---|---|---|
| Person1 | 0.740 | OR(AND(doublemeta(x.surname,y.surname)\|1.0,caverphone2 (x.surname,y.surname) \|1.0)\|0.0,matchrating(x.surname,y.surname)\|1.0) |
| Person2 | 0.500 | OR(doublemeta(x.surname,y.surname)\|1.0,soundex (x.surname,y.surname)\|1.0) |
| Restaurants | 0.300 | AND(doublemeta(x.name,y.name)\|0.78,soundex(x.name,y.name)\|0.73) |
| DBLP–ACM | 0.900 | cologne(x.title,y.title)\|1.0 |
| ABT–BUY | 0.350 | OR(cologne(x.description,y.name)\|0.48,matchrating(x.name,y.name)\|1.0) |
| Drugs | 1.000 | AND(AND(doublemeta(x.name,y.name)\|1.0,caverphone2 (x.name,y.name)\|1.0)\|0.0,nysiis(x.name,y.name)\|1.0) |
| Amazon–GoogleProducts | 0.006 | MINUS(matchrating(x.description,y.name)\|0.66,soundex (x.title,y.name)\|1.0) |

Table 8: Hybrid LS generated using Wombat algorithm with respect to *F-Measure* (F).

| Datasets | F | LS |
|---|---|---|
| Person1 | 0.870 | AND(jaccard(x.surname,y.surname)\|1.0,soundex(x.given-name,y.given-name)\|1.0) |
| Person2 | 0.540 | OR(AND(jaccard(x.surname,y.surname)\|1.0,doublemeta(x.surname,y. surname)\|1.0)\|0.0,soundex(x.surname,y.surname)\|1.0) |
| Restaurants | 0.460 | OR(caverphone1(x.name,y.name)\|0.81,cosine(x.name,y.name)\|0.53) |
| DBLP–ACM | 0.910 | OR(refinedsoundex(x.title,y.title)\|1.0,cologne(x.title,y.title)\|1.0) |
| ABT–BUY | 0.290 | OR(jaccard(x.name,y.name)\|0.43,cologne(x.description,y.name)\|0.48) |
| Drugs | 1.000 | AND(match(x.name,y.name)\|1.0,caverphone2(x.name,y.name)\|1.0) |
| Amazon–GoogleProducts | 0.450 | qgrams(x.title,y.name)\|0.43 |

We plan to develop machine learning techniques that consider the nature of phonetic algorithms (e.g., the length of code represents the encoded terms) in a way that machine learning algorithms can decide to use either phonetic similarity measures or string similarities based on the given dataset and the length of the encoded terms . We will also study the impact of phonetic algorithms in LD with respect to run times. The explainability of generated hybrid LS in terms of explainable AI will be ensured by extending the LS explanation approach introduced in [2] to deal with phonetic similarities.

## ACKNOWLEDGMENTS

This work has been supported by the BMVI projects LIMBO (GA no. 19F2029C) and OPAL( no. 19F2028A), Eurostars Project SAGE (GA no. E!10882) as well as the H2020 project SLIPO (GA no. 731581).

## REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. *SIGMOD Rec.* 22 (June 1993), 207–216. Issue 2. https://doi.org/10.1145/170036.170072

[2] A F Ahmed, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. 2019. LSVS: Link Specification Verbalization and Summarization. In *24th International Conference on Applications of Natural Language to Information Systems (NLDB 2019) (Lecture Notes in Computer Science)*.

[3] Aw AiTi, Zhang Min, Yeo PohKhim, Fan ZhenZhen, and Su Jian. 2005. Input normalization for an english-to-chinese sms translation system. In *The Tenth Machine Translation Summit*.

[4] Arup Kumar Bhattacharjee, Atanu Mallick, Arnab Dey, and Sananda Bandyopadhyay. 2013. Enhanced Technique for Data Cleaning in Text File. *International Journal of Computer Science Issues (IJCSI)* 10, 5 (2013), 229.

[5] Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *KDD*. 39–48.

[6] Moisés G. Carvalho, Albero H. F. Laender, Marcos André Gonçalves, and Altigran S. da Silva. 2008. Replica identification using genetic programming. ACM.

[7] Peter Christen. 2006. A comparison of personal name matching: Techniques and practical issues. In *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)*. IEEE, 290–294.

[8] Peter Christen. 2008. Febrl -: an open source data cleaning, deduplication and record linkage system with a graphical user interface. In *KDD '08*. 1065–1068.

[9] Nello Cristianini and Elisa Ricci. 2008. Support Vector Machines. In *Encyclopedia of Algorithms*.

[10] Yerai Doval, Manuel Vilares, and Jesús Vilares. 2018. On the performance of phonetic algorithms in microtext normalization. *Expert Systems with Applications* 113 (2018), 213 – 222. https://doi.org/10.1016/j.eswa.2018.07.016

[11] Carmen Gálvez. 2006. Identificación de nombres personales por medio de sistemas de codificación fonética. *Encontros Bibli: revista eletrônica de biblioteconomia e ciência da informação* 22 (2006), 105–116.

[12] Wael H Gomaa and Aly A Fahmy. 2013. A survey of text similarity approaches. *International Journal of Computer Applications* 68, 13 (2013), 13–18.

[13] David Holmes and M Catherine McCabe. 2002. Improving precision and recall for soundex retrieval. In *Proceedings. International Conference on Information Technology: Coding and Computing*. IEEE, 22–26.

[14] Robert Isele and Christian Bizer. 2011. Learning Linkage Rules using Genetic Programming. In *Sixth International Ontology Matching Workshop*.

[15] R. Isele, A. Jentzsch, and C. Bizer. 2011. Efficient Multidimensional Blocking for Link Discovery without losing Recall. In *WebDB*.

[16] Robert Isele, Anja Jentzsch, and Christian Bizer. 2012. Active Learning of Expressive Linkage Rules for the Web of Data. In *Web Engineering*, Marco Brambilla, Takehiro Tokuda, and Robert Tolksdorf (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 411–418.

[17] Lianyin Jia, Lulu Zhang, Guoxian Yu, Jinguo You, Jiaman Ding, and Mengjuan Li. 2018. A Survey on Set Similarity Search and Join. *International Journal of Performability Engineering* 14, 2 (2018).

[18] S. Sathiya Keerthi and Chih-Jen Lin. 2003. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Comput.* 15 (July 2003), 1667–1689. Issue 7. https://doi.org/10.1162/089976603321891855

[19] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2009. Comparative evaluation of entity resolution approaches with FEVER. *Proc. VLDB Endow.* 2, 2 (2009), 1574–1577.

[20] Odell M. and Russell R.C. 1918. The Soundex coding system. (1918).

[21] Gary Mokotoff. 2007. Soundexing and genealogy. *URL: http://www. avotaynu. com/soundex. html* (2007).

[22] Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. 2017. A survey of current link discovery frameworks. *Semantic Web* 8, 3 (2017), 419–436.

[23] Axel-Cyrille Ngonga Ngomo and Sören Auer. 2011. LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In *IJCAI*.

[24] Axel-Cyrille Ngonga Ngomo, Jens Lehmann, Sören Auer, and Konrad Höffner. 2011. RAVEN: Active Learning of Link Specifications. In *Proceedings of the Ontology Matching Workshop (co-located with ISWC)*. Springer. http://jens-lehmann.org/files/2011/raven.pdf

[25] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. 2012. EAGLE: Efficient Active Learning of Link Specifications Using Genetic Programming. Springer Berlin Heidelberg.

[26] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. 2013. Unsupervised learning of link specifications: deterministic vs. non-deterministic. In *Proceedings of the Ontology Matching Workshop*.

[27] Axel-Cyrille Ngonga Ngomo, Klaus Lyko, and Victor Christen. 2013. COALA – Correlation-Aware Active Learning of Link Specifications. In *Proceedings of ESWC*. Springer.

[28] Andriy Nikolov, Mathieu d'Aquin, and Enrico Motta. 2012. Unsupervised learning of link discovery configuration. In *The Semantic Web: Research and Applications*. Springer, 119–133.

[29] Vimal P Parmar and CK Kumbharana. 2014. Study Existing Various Phonetic Algorithms and Designing and Development of a working model for the New Developed Algorithm and Comparison by implementing it with Existing Algorithm (s). *International Journal of Computer Applications* 98, 19 (2014), 45–49.

[30] Lawrence Philips. 1990. Hanging on the Metaphone. *Computer Language Magazine* 7, 12 (December 1990), 39–44. Accessible at http://www.cuj.com/documents/s=8038/cuj0006philips/.

[31] Lawrence Philips. 2000. The double metaphone search algorithm. *C/C++ users journal* 18, 6 (2000), 38–43.

[32] David Pinto, Darnes Vilarino, Yuridiana Alemán, Helena Gómez, and Nahun Loya. 2012. The soundex phonetic algorithm revisited for sms-based information retrieval. In *II Spanish Conference on Information Retrieval CERI*.

[33] Hans Joachim Postel. 1969. Die Kölner Phonetik. Ein Verfahren zur Identifizierung von Personennamen auf der Grundlage der Gestaltanalyse. *IBM-Nachrichten* 19 (1969), 925–931.

[34] Rima Shah and Dheeraj Kumar Singh. 2014. Analysis and comparative study on phonetic matching techniques. *International Journal of Computer Applications* 87, 9 (2014).

[35] Mohamed Ahmed Sherif, Axel-Cyrille Ngonga Ngomo, and Jens Lehmann. 2017. WOMBAT - A Generalization Approach for Automatic Link Discovery. Springer.

[36] Chakkrit Snae. 2007. A comparison and analysis of name matching algorithms. *International Journal of Applied Science. Engineering and Technology* 4, 1 (2007), 252–257.

[37] Dezhao Song and Jeff Heflin. 2011. Automatically Generating Data Linkages Using a Domain-Independent Candidate Selection Approach. Springer Berlin Heidelberg.

[38] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. 2011. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *PVLDB* 5, 3 (2011), 157–168.

[39] Minghe Yu, Guoliang Li, Dong Deng, and Jianhua Feng. 2016. String similarity search and join: a survey. *Frontiers of Computer Science* 10, 3 (2016), 399–417.

[40] Yufei Yuan and Michael J. Shaw. 1995. Induction of fuzzy decision trees. *Fuzzy Sets Syst.* 69 (January 1995), 125–139. Issue 2. https://doi.org/10.1016/0165-0114(94)00229-Z

[41] Justin Zobel and Philip Dart. 1996. Phonetic string matching: Lessons from information retrieval. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 166–172.