Linking Graph Entities with Multiplicity and Provenance

Jixue Liu
University of South Australia
Adelaide, Australia
jixue.liu@unisa.edu.au

Selasi Kwashie*
Data61 - CSIRO
Adelaide, Australia
selasi.kwashie@data61.csiro.au

Jiuyong Li
University of South Australia
Adelaide, Australia
jiuyong.li@unisa.edu.au

Lin Liu
University of South Australia
Adelaide, Australia
lin.liu@unisa.edu.au

Michael Bewong
Charles Sturt University
Wagga Wagga, Australia
mbewong@csu.edu.au

ABSTRACT

Entity linking and resolution is a fundamental database problem with applications in data integration, data cleansing, information retrieval, knowledge fusion, and knowledge-base population. It is the task of accurately identifying multiple, differing, and possibly contradicting representations of the same real-world entity in data. In this work, we propose an entity linking and resolution system capable of linking entities across different databases and mentioned-entities extracted from text data. Our entity linking/resolution solution, called Certus, uses a graph model to represent the profiles of entities. The graph model is versatile, thus, it is capable of handling multiple values for an attribute or a relationship, as well as the provenance descriptions of the values. Provenance descriptions of a value provide the settings of the value, such as validity periods, sources, security requirements, etc. This paper presents the architecture for the entity linking system, the logical, physical, and indexing models used in the system, and the general linking process. Furthermore, we demonstrate the performance of update operations of the physical storage models when the system is implemented in two state-of-the-art database management systems, HBase and Postgres.

KEYWORDS

entity resolution, entity linking, graph data, graph model, provenance, multiplicity, text data

1 INTRODUCTION

In entity linking and resolution, *entities* refer to real-world objects (e.g., people, locations, vehicles, etc.) and real-world happenings (e.g., events, meetings, interactions, etc.). Entities are described by data in information systems. However, the descriptions may be repeated and different in these systems. In a database, for instance, a person may have more than one record in a table, and the records may have repeating, differing and contradicting information about the person. Likewise, two different databases may capture different information about the same entity. For example, a medical database only concerns with a person's health related properties, whereas an immigration database only concerns with the truthfulness of a person's identity.

*Corresponding author.

A description of an entity is called a *profile*, and it can be a record in a relational database or a paragraph of words about an entity in a document. An entity may have multiple profiles in one or more sources. In other words, multiple profiles in one or more databases (or documents) may refer to the same real-world entity.

Once the profiles of entities are captured into a database, the profiles and the entities become separated in that the users of the database know the profiles, but possibly, not the entities. This separation raises a serious issue. Answering the question of whether a given profile refers to a particular real-world entity is non-trivial and challenging. For example, given the profile: `{name: Michael Jordan, nationality: American, occupation: athlete}`, there are at least four real-world persons whose profiles in Wikipedia match this description (see [1] for details). A dual problem to the above problem is whether two profiles, which may look similar or very different, refer to the same real-world entity. This dual problem is as hard as the above problem.

The goal of entity linking research is to design methods to derive an answer to the dual question: *do a pair of given profiles refer to the same entity*? When a pair of profiles are found to refer to the same entity, one of two actions may be taken. One is to remove of one of the profiles. This is called *deduplication*. The other is to merge the two profiles and this is called *resolution/linking*[1].

Three complications make the linking/deduplication task more difficult. The first complication is from *non-alignment of attributes and relationships*. That is, different profiles describe entities using different attributes and/or relations. This is illustrated by the profiles p_1 and p_2 in Table 1. The two profiles have different attributes except for the `name` attribute. The non-aligned attributes make their match less possible. The second complication is from the *multiplicity of values*. For example, compared with the profile p_1, the profile p_3 has two `name` values. The third complication is the presence of *provenance data*. Provenance data describes the background information of a value as well as the validity period(s), security & access restriction(s), source(s), etc., of a value. For example, in p_4, `{since 2005}` specifies when the `name` 'George' started being used, and `{2010}` indicates when the `height` valued '160' was taken. Unlike non-alignment, multiplicity and provenance of values can be useful as they provide more information. However, their usefulness comes

[1]We consider linking and resolution, and use the terms interchangeably in this work

Table 1: Complexity of entity profiles

p_1	`{name:George, birth-date:1/Jan/2000}`
p_2	`{name:George, sex:male, height:160}`
p_3	`{name:George, name:Jord, sex:male, height:160}`
p_4	`{name:George {since 2005}, name:Jord, height:160{2010}}`

at a cost: they require more powerful matching algorithms and data structures to enable effective usage.

This paper presents the system supporting our entity linking method Certus [11], the data and index models that enable multiplicity and provenance of attribute and relationship values to be accurately captured and leveraged for effective and efficient entity linking. The contributions of the paper are as follows.

- First, we present the architecture of our entity linking system (Section 3). This architecture enables textual data to be processed and the entities described in the texts can be linked to entity profiles from other data sources. The architecture uses Elasticsearch[2], an index engine, to increase the linking and search efficiencies.
- Secondly, we propose a graph model for entity linking involving multiplicity of attribute and relation values with provenance information (subsection 4.1). In this model, the attributes and relations of profiles are well-represented by lists of sets (of attribute/relation, value, and provenance), instead of dictionaries of attribute- and relation-value pairs. Our model enables provenance and value-multiplicity to be captured, indexed and used correctly.
- Thirdly, we propose physical models for the storage of the graph of entity profiles; detail the index structures that support effective search and blocking operations (subsections 4.2 & 4.3); and give the processes in the entity linking component of the system (Section 5).
- Lastly, we show experiments about the time performances of our physical model implementations on both relational and non-relational database management systems (Section 6).

2 RELATED WORK

Entity linking and resolution is a well-known database problem that has attracted volumes of research in the literature, especially in the relational data setting. Readers are referred to [14] for details. In general, the existing works focus on two main directions: accuracy and efficiency. The accuracy concern is on finding true matches of different entity profiles when they refer to the same real-world entity without introducing false matches. A more specific term called *efficacy* is defined to mean accuracy in [2]. The efficiency issue is about alleviating the infeasible pairwise comparison of profiles, and making the linking process scalable in large data.

For accurate entity linking and deduplication, early works on the subject examined many methods such as cosine similarity match, distance-based match, TF/IDF, and Soundex. The well known similarity measures for entity linking are summarized and reviewed in [10]; and the work in [9] presents a comparative evaluation of some existing works.

The efficiency problem has also drawn significant research attention. The complexity of calculating the exact similarity between profile pairs is $O(n^2)$. Given a large number of entity profiles, say $n = 100$ million, the time for computing similarity is too long to be practical. Thus, several ideas have been introduced in the literature to address the problem, like canopy (sorting and moving window), hierarchical, bucketing (clustering), and indexing approaches. In practice, the indexing approach has been found to be more useful, resulting in the proposal of a plethora of indexing methods in the literature (see [4, 17] for surveys of techniques).

In the recent years, there has been an increasing research interest in linking entity-mentions in texts to existing entities in knowledge-bases. From Wiki Miner in [13], many works have been produced in this area and are reviewed in [5, 20]. The fundamental steps in text-based linking include: entity-mention detection, candidate matching-entity generation, and candidate matching-entity ranking. The work in [5] reviews the methods for detecting entity-mentions in texts. Whereas the review paper [20] summarizes the details of how features (such as the mentions, types, contexts, etc.) and models (e.g., unsupervised, supervised, probabilistic, graph-based, and combined methods) are used in the ranking of candidate matching-entities. The efforts toward ranking is continuing, and the work in [12] aims to identify effective relationship words among entity-mentions to increase the accuracy of linking.

Most data management and software companies claim to support entity linking in structured data, but the systems are often not available for evaluation. In contrast, a number of open source research frameworks are available on entity linking in text data. For example, [13] proposes a method to extend terms in texts using Wikipedia pages. [6] is a framework tagging terms in *short* texts by Wikipedia pages, which is then followed by the works in [8, 19] for software improvement. [19] and [3] are other tools that contain a three step implementation for linking entity-mentions in text to Wikipedia pages. The work in [22] sets up a framework for entity linking work to be tested and evaluated.

There exists works in the literature on the support and use of provenance for entity linking. For example, [15] is on provenance modeling and capture for entity linking whereas [23] presents a provenance-aware framework for improving entity linking results. Our work models, supports, and leverages provenance as well as attribute- and relation-value multiplicity for accurate entity linking in both structured and text data.

3 SYSTEM ARCHITECTURE & FUNCTIONS

This section covers the architecture of our entity linking system, and outlines the functions of the components of the system.

Figure 1 presents an overview of the architecture of our system. Central in the system is the Knowledge-Base (KB) which is a graph of entity profiles (details in Section 4). The profiles in the KB come from three sources: (a) ingested profiles from different data sources (through the *Ingester*) with no restriction on model; (b) extracted profiles from user-supplied textual documents (via the *Text Parser*); and (c) profiles created from the *User-Interface* (UI). The profiles are linked and indexed by the *Entity Linking & Resolution* (ELR) and *Indexer* components respectively. And, all user interactions with the system are via the UI, mediated by the *Query Processor.*

The following are brief details and functions of the components.

[2]https://www.elastic.co/

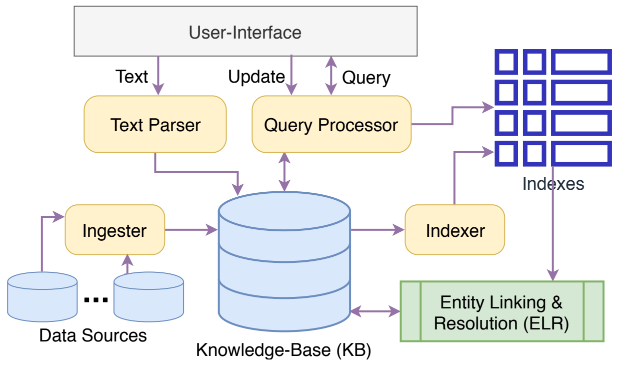


Figure 1: The architecture of our entity linking system

The Ingester: maps entity descriptions from various data sources into graph-modelled profiles in the knowledge-base. Its operation is straight-forward and dependent on the respective models (or lack thereof) of the various sources of data.

The Text Parser: reads textual user-inputs (e.g., documents, reports, etc.), and extracts mentioned-entities and their relationships from the texts, and stores the extracted entity profiles into the knowledge-base. In our implementation, we use Stanford NER [7], Stanford POS tagger [21], and Open IE 4.x [16] for this purpose.

The problem with the above-mentioned packages is that they may produce many triples (subject, relation, object) that do not reflect the original intention of authors in the writings. For example, extractions for the sentence "*John said that, Peter has taken away the mobile phone*", include the triple: (*"Peter", "has taken away", "the mobile phone"*). This extract is only syntactically correct. The semantic correctness of this extraction is, however, dependent on John's credibility/position. If John is a Police spokesman, for instance, then the chance of semantic-correctness would be high. However, if John is an adversary of Peter, for example, then the chance for the extraction to be correct would be low.

Therefore, we developed some heuristic rules to filter ambiguous extractions. The rules: (a) replace coreferences (pronouns) with the actual entity-mentions; (b) remove extractions that are conditional, and indirect speech; and (c) filter extractions that describes feelings and emotions. The inputs to the rule-based filtering system (RbFS) are the text and labelling from Stanford NLP. The rules improve the F1-score of extractions by 18% on average on our test datasets. The details of the RbFS is out of the scope of this paper.

The Query Processor: receives requests from the user-interface and responds based on the request type. An insert or update request is directly sent to the knowledge-base. For a query by profile identifier or keywords, the index is searched and then answers are retrieved from the knowledge-base.

The Indexer: keeps the indexes up to date with the current system state. Whenever the knowledge-base is updated, this component sends the update to the indexes. The indexes support users' queries and the ELR component. We use Elasticsearch, an open source distributed search and index engine, as our index management system. Later on (in subsection 4.3), we present details of the structure of the indexes in our system.

The ELR component: as the name suggests, is the main component in the system. In principle, for every entity profile p in the knowledge-base, indexes are read for candidate matching profiles; calculations of the similarity of p to each of the candidates are performed; and the knowledge-base is updated to store the similarities. A candidate p' of p from the indexes is a profile that is roughly similar to p, i.e., the pair share some similar attribute and/or relationship values. We remark that the fact that p and p' share a similar 'word' does not necessarily mean that they refer to the same real-world entity. For instance, if p is a `male` with the name `Pete` and p' is a `female` and has a `friend` called `Peter`, then p' can be a candidate of p as they share a similar value (i.e., `Pete & Peter`), but they do not match. Therefore, the indexes merely give a set of possible profiles that may match which require further evaluation.

4 MODELS

In this section, we present the logical, physical, and indexing models used in our entity linking system.

4.1 Modelling of Profiles

A real-world entity, naturally, has many attributes (or properties) and relates to other entities in multiple ways. An entity profile (simply, profile) captures some of the attributes and relationships of an entity; and another profile may capture different attributes and relationships of the same entity with possible overlaps and contradictions. For example, a person may have multiple profiles in the same/different sources of data (e.g., databases, knowledge-bases, social networking sites, etc.).

Entity profile structure. The data structure of profiles should be able to capture multiple values of the same attribute or relationship as well as their provenance information. This is because of the ever-changing or evolving nature of the properties and relationships of real-world entities. For example, a person may change his/her names, live at different addresses over time, have multiple marriages spanning different periods, etc. These changes lead to multiple values for attributes and relations and these values may be associated with provenance information.

We represented an entity profile as a triple $p = \langle id, \mathcal{A}, \mathcal{R} \rangle$, where: id is the identifier of the profile, $\mathcal{A} = [a_1, \cdots, a_n]$ is a list of attribute-objects, $\mathcal{R} = [r_1, \cdots, r_m]$ is a list of relationship-objects; and each $a \in \mathcal{A}$, $r \in \mathcal{R}$ is a set of ordered[3] key-value pairs. Four profile examples are given in Table 2, shown in our structure. Profile p_1 describes a `person` entity: a `male` called `Peter` up to `1991` and now called `John`. He `lived_at` location `L1` from `1989` to `1995`, `owns` `L1` since `1989` and has a `friend` named `Bob`. Our data structure for profiles is thus able to capture the multiplicity and provenance of values.

Entity profiles graph. We use a graph model for modelling profiles, as it is capable of representing any number of attributes and relationships. Moreover, since $\mathcal{A}$ and $\mathcal{R}$ are defined as lists, instead of dictionaries, value multiplicity can be presented easily. Furthermore, the edges of the profiles graph allow the traversal of the profiles.

Formally, we use the following definition of an entity profiles graph, $G = (V, E, F_{\mathcal{A}})$, where: (i) V is a finite set of nodes; (ii) E is a finite set of edges, given by $E \subseteq V \times V$; (iii) each node $v \in V$ (resp.

[3]attribute/relation key-value pairs are first in the set, followed by the provenance data (if exists)

Table 2: Example of profiles as triples

p_1	⟨ P1, $\mathcal{A}$ = [{type: person}, {name: John}, {name: Peter, until: 1991}, {sex: m}], $\mathcal{R}$ = [{lives_at: L1, from: 1989, to: 1995}, {friend: P2}, {owns: L1}] ⟩
p_2	⟨ P2, $\mathcal{A}$ = [{type: person}, {name: Bob}, {name: John, until: 1990}, {bdate: 1980.12.12 }], $\mathcal{R}$ = [{lives_at: L1, from: 1990, to: 2000}, {lives_at: L2, from: 2001}, {friend:P1}] ⟩
p_3	⟨ L1, $\mathcal{A}$ = [{type: location}, {numb: 1}, {street: Brown Blvd.}, {post:2000}], $\mathcal{R}$ = [{owned_by: P1, from: 1989}] ⟩
p_4	⟨ L2, $\mathcal{A}$ = [{type: location}, {numb: 69}, {street: Brown Ave.}, {post:5000}] ⟩

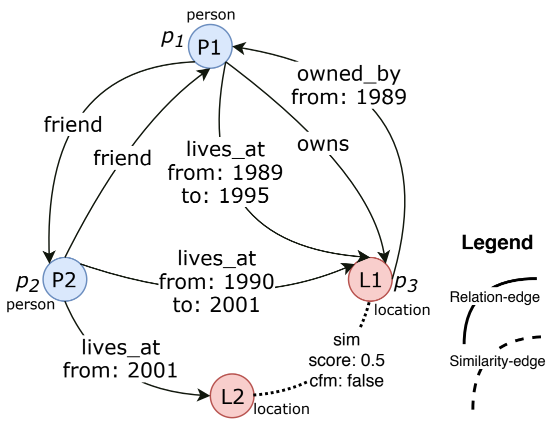


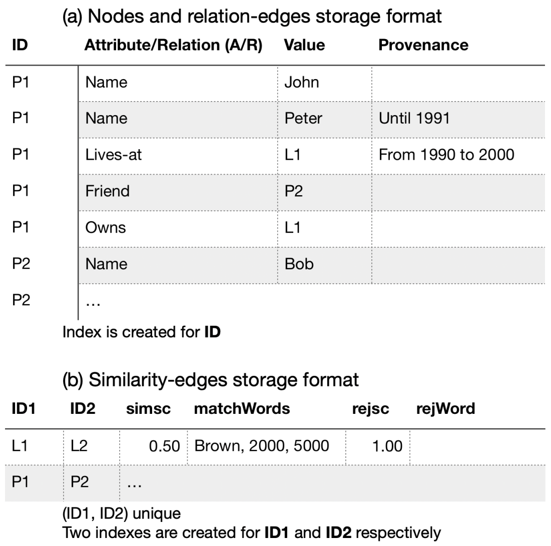
Figure 2: Graph representation of profiles in Table 2

edge $e \in E$) has a label $L(v)$ (resp. $L(e)$); and (iv) each node $v \in V$ has an associated list $F_{\mathcal{A}}(v) = [a_1, \cdots, a_n]$ of attribute-objects.

A node in an entity profiles graph represents a profile p, identified by the profile *id*, and is associated with $\mathcal{A}$ and $\mathcal{R}$ as defined. Two types of edges exist in the graph. One is called a *relation-edge*, derived from $\mathcal{R}$. That is, the edge $(rel, P1, P2)$ is an edge in the graph iff.: $P2$ is the value for relation rel in $P1$ where $P1$ and $P2$ are profile/node identifiers. The second type of edge is called a *similarity-edge*, derived from the profile pair similarity and has the form $(sim, P1, P2, score, cfm)$ where sim is a fixed label, $score$ is the similarity score (defined later in Section 5), and cfm is a binary indicator showing whether the link-state of a profile pair has been confirmed by a user. The indicator is necessary because, in sensitive systems like policing, we want 100% precision if two profiles are linked. Thus, cfm requires user-interaction (to be discussed further in Section 5). Figure 2 is an example of the graph of profiles in Table 2. Note that each node in the graph carries its attribute list (not shown in the diagram).

4.2 Physical Model for Profiles

Profiles are modelled as a graph; and the nodes and relation-edges are stored in one structure while the similarity-edges are stored in a separate structure. Similarity-edges are updated frequently as any profile change triggers a re-computation of similarities for the profile and other affected profiles. Therefore, storing similarity-edges in a separate structure improves the update efficiency. The two structures for storing the graph are called the *physical model* and shown in Figure 3.



(a) Nodes and relation-edges storage format

ID	Attribute/Relation (A/R)	Value	Provenance
P1	Name	John	
P1	Name	Peter	Until 1991
P1	Lives-at	L1	From 1990 to 2000
P1	Friend	P2	
P1	Owns	L1	
P2	Name	Bob	
P2	…		

Index is created for **ID**

(b) Similarity-edges storage format

ID1	ID2	simsc	matchWords	rejsc	rejWord
L1	L2	0.50	Brown, 2000, 5000	1.00	
P1	P2	…			

(ID1, ID2) unique
Two indexes are created for **ID1** and **ID2** respectively

Figure 3: Physical model for storing the profiles graph

The model in Figure 3 is self-explainable; and the data in the two tables of the model are derived from some of the exemplar profiles in Table 2. The table in Figure 3(a) stores the nodes (profile ids and attributes) and relation-edges (relationships). Each node uses multiple lines and each line is for an attribute or relationship value pair with provenance details. The table in Figure 3(b), on the other hand, is for the storage of similarity-edges and each pair of profiles has an entry in the structure. *simsc* and *rejsc* represent similarity score and rejecting score respectively (details in Section 5). Since the size of the table in Figure 3(b) is the square of the number of nodes/profiles, to reduce the size, a threshold may be used to filter out very lowly-scored entries.

We realize that the performance of accessing the similarity-edge table plays a crucial role in the overall linking time performance due to its frequent update operations. Therefore, we show empirical results on three different implementation options of the physical model in Section 6.

4.3 Index Mappings

Our aim is to design index structures to support users' search for profiles and support the candidate matching-profiles generation (*a.k.a* blocking) of the ELR component. Thus, we use Elasticsearch, a distributed index management system that can support multiple indexes with various structures.

Recall that, logically, each profile is a triple of the form $\langle id, \mathcal{A}, \mathcal{R} \rangle$ in the knowledge-base. We consider two options for building indexes for the profiles (discussed below), and both are configured with the double-metaphone phonetic analyzer [18], and custom-built synonym and alias transformers.

Keyword search & blocking indexes. The indexes that support keyword search of profiles and blocking for entity linking uses a set of words generated from the profiles. The set of words are values from the profile without provenance. That is, the provenance values, the structure, the attribute and relation names are all ignored, relationship targets (i.e., other profile ids) are replaced by the summary of the target, and all duplicate words are removed. For example, the target summary for p_1 in Table 2, is a bag of the following values: "`John, Peter, m, 1, brown, 2000`".

The 'loose' structure of these indexes guarantee high recall of search and blocking results.

Structured search indexes. The indexes for structural search consider the structure of the profiles. For example, if a user wants to find a person with "`name : John, lives_at : 1 Brown street − {until : 2000}`", the index should enable p_1 in Table 2 to be found. To support such structural search, we build indexes with the *nested mappings* in Elasticsearch for profiles structured as JSON objects with:

$$\mathcal{A} = [\{\mathtt{A_1 : v_1, from : t_0, to : t_1}\}, \{\mathtt{A_1 : v_2, to : t_2}\}, \{\mathtt{A_2 : v_3}\}, \cdots],$$

where t_0, t_1, t_2 are date time values; $\mathcal{R}$ mapping similarly defined.

We remark that the usage of nested mappings is critical to the preservation of the correct semantics of the multiplicity of values and their associated provenance information in Elasticsearch. Otherwise, Elasticsearch indexes the profiles in a 'flat-format' of the form: $\{\mathtt{A_1 : [v_1, v_2], A_2 : [v_3], from : [t_0], to : [t_1, t_2]}\}$ – which loses semantics and leads to errors and very low precision.

In settings where smaller and more precise blocking are required, the nested-mapped indexes should be used.

5 LINKING OF PROFILES

This section presents a description of our profiles comparison and linking processes. First, we highlight some relevant preprocessing steps. Then, we detail the profile-pair comparison and evaluation; and finally, give a brief overview of the match prediction and confirmation.

Preprocesses. Prior to the calculation of the similarity between profile pairs, some preprocessing are necessary. For example, consider person and location entities: it is important to tackle the disparate representation of the same names and addresses respectively. The name `Richard` is often aliased as `Dick`; and the street-type `Boulevard` is often shortened as `BLVD`. To enable Dick to match Richard, a dictionary of name aliases of people is created (similarly, for addresses). Each name/address in a profile is checked against the dictionary. If the name has an alias, the name is expanded in the form "name alias", e.g., "Richard Dick". Similar operations are performed on the initials, and pre-/post-fixes of names.

Similarity evaluation. Given two profiles p_1 and p_2, our entity linking method uses two scoring and one decision processes to determine whether they refer to the same entity in the real-world. The two scores are the similarity score, *simsc*, and the rejection score, *rejsc*; while the decision process is a data-dependency-based prediction model. We discuss the scoring here.

Given a profile p, we use the notation $X \in p$ to represent either an attribute X in $p[\mathcal{A}]$ or a relation X in $p[\mathcal{R}]$. The similarity score, *simsc*, of two profiles p_1, p_2, is calculated as follows:

$$simsc(p_1, p_2) = \sum_{X \in p_1, p_2} \mathcal{M}(p_1.X,\ p_2.X) \cdot \mathcal{I}_X(p_1.X, p_2.X),$$

where $\mathcal{M}$ is a function that returns a value indicating the level of approximate match between a pair of values for the same attribute/relation X, and $\mathcal{I}$ returns the level of information supplied by the match $\mathcal{M}$ for the values of X.

The function $\mathcal{M}$ considers many factors, dependent on the attribute/relation; and the values of an attribute/relation are in the form of a bag of words after synonym/alias expansion with provenance data. For example, to evaluate a `name` match for person entities, $\mathcal{M}$ considers the initials, ordering, post-/prefixes, aliases, and phonetics of names, as well as n-gram matching of character/word sequences. Edit distance is used after n-gram matching to improve accuracy and efficiency. If two values match within a user-specified threshold, then the provenance information are considered.

The function $\mathcal{I}$ returns the highest *information level* of matching values. For example, for the name-pair "`John Smith White`" and "`Jones Smiths Green`", the $\mathcal{I}_{name}$-weight is derived as:

$$max\{\frac{inf(John) + inf(Jones)}{2}, \frac{inf(Smith) + inf(Smiths)}{2}\}.$$

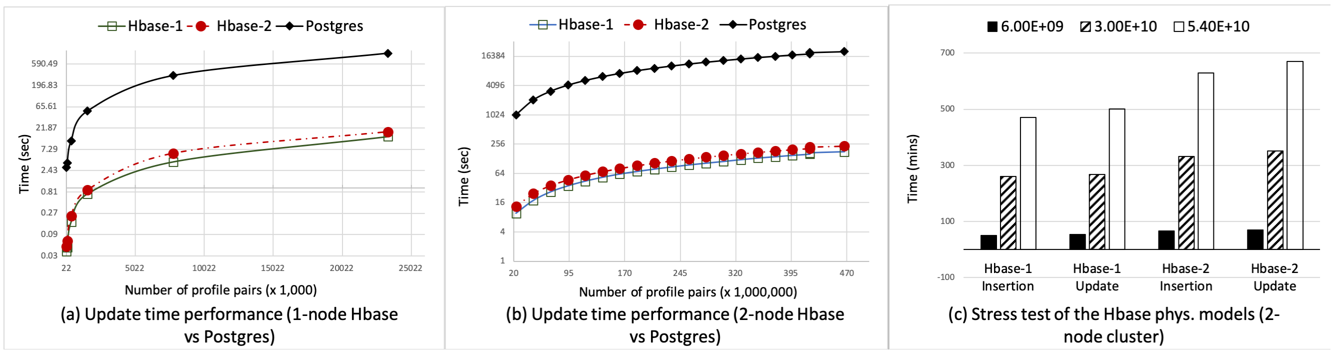
The function $inf(w)$ indicates the probability of two profiles to be linked if they match on the value w. Note that, in this example, the name-pair "Green" and "White" are not considered in the evaluation of $\mathcal{I}$ as they are dissimilar (i.e., have low $\mathcal{M}$ value). Intuitively, if a word w is rare, it has high $inf(w)$-value. Consider the two first-names 'John' and 'Cherith'. When two profiles share the name 'Cherith', the probability for the two profiles to be linked is much higher than when two profiles share the name 'John'.

The $inf(w)$-value of a word w is controlled by two factors: the number $m(w)$ of profiles sharing the word, and the number $k(w)$ of real-world entities shared by the profiles sharing the word. If $k(w)$ is large, the fact that the $m(w)$ profiles share the same word contributes very little to the linking, and $inf(w)$ should be small. When the total number of profiles increases, the chance for two profiles to share the same word becomes larger but the crucial control of the probability is still by k/m. That is, $inf(w) \propto m/k$; and m can be easily obtained from word statistics but k is often unknown. A large m does not mean a large m/k ratio. We use a variation of the Sigmoid function to estimate $inf(w)$, given as:

$$inf(w) = \frac{1}{1 + exp(\alpha \cdot m(w) - \beta)},$$

where α, β control the steepness of the decay curve and its midpoint respectively. For a given w, $k(w)$ can be empirically estimated and linked to β. However, in general, our empirical results suggest $\alpha = 0.1$ and $\beta = 60$ are suitable settings for our applications.

The *rejsc* score, on the other hand, is based on a simple penalty system. Given two profiles with a high overall *simsc* score, a penalty of 1 is added to their *rejsc* score if the pair are dissimilar on a *key*



(a) Update time performance (1-node Hbase vs Postgres)

(b) Update time performance (2-node Hbase vs Postgres)

(c) Stress test of the Hbase phys. models (2-node cluster)

Figure 4: Time performance of update transactions

Implementation	Schema
(a) Postgres	SimTable = (PID1, PID2, simsc, matchWords, rejsc, rejWords) *indexes on PID1 and PID2 resp.*

(b) HBase-1	HTable1 = (pid-pair : CF(simsc, matchWords, rejsc, rejWords))
	HTable2 = (PID2 : CF(PID1, pid-pair))
	each profile has an entry in HTable1 and an entry in HTable2

(c) HBase-2	HTable3 = (pid-pair : CF(simsc, matchWords, rejsc, rejWords)) *each profile has two entries with the same CF*

Figure 5: Implementation options of the similarity structure

attribute/relation (determined by application and domain). For example, in a law enforcement context, one such key attribute for person and location entities is `birth-date` and `zip-code` respectively.

Match prediction & confirmation. As mentioned earlier, we use a data-dependency aided decision model to predict whether a given pair of similar profiles refer to the same real-world entity. This decision model is a major topic (and we refer interested readers to the paper on it in [11]). The approach eliminates the challenge of and need for fine-tuning dis/similarity thresholds for approximate matching, through the use of a discovery algorithm that learns matching rules in labeled data. The match prediction model achieves high precision without significant compromise of recall.

In some applications, even accurate prediction of the linked status of two profiles require human confirmations. Thus, our entity linking system supports this scenario, allowing the keeping of domain experts in the loop. Indeed, every similarity-edge between profiles carry the data structure for the confirmation of predicted matches (when needed). For example, in Figure 2, the similarity-edge between nodes `L1` and `L2` is not confirmed (i.e., `cfm: false`).

6 EXPERIMENTS

In this section, we empirically evaluate the performance of the three different implementations of the physical model. We remark that, the accuracy of the ELR system is already evaluated in [11].

We note that updating the pairwise similarities of profiles is a major performance bottleneck. This is because, for every 1,000 profiles, the updated similarity entries are around 20,000-100,000. Therefore, we examine the time efficiency of accessing the similarity structure (Figure 3(b)).

All procedures in the work are implemented in Java, and the entity linking system runs on Ubuntu 18.04 machine(s). For single-machine tests, the experiments were run on an Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz computer with 32GB of memory. In the cases where multi-node HBase clusters are required, an Intel(R) Core(TM) i7 CPU @ 2.30GHz computer with 16GB of memory is added. The versions of Postgres and HBase used are 9.6.12 and 1.4.8 respectively.

Efficiency of Similarity Storage Structure

We present our experiment results on the efficiency of accessing the similarity structure on different platforms with different implementations. We tested the implementation in Postgres[4] and HBase, with schemas summarized in Figure 5.

The operations to access the similarity structure include search, insertion, update and deletion. Since the similarity is for a pair, the search must be supported from either ID. We created two indexes for this purpose in the relational option of Postgres. With the HBase options, CF means a column family which is a dictionary of key-value pairs with the keys listed in the brackets. The 'id-pair' is constructed by ID1+"-"+ID2. In the case of HTable3 in Figure 5(c), the second id-pair is ID2+"-"+ID1.

Performance of the physical model on small to large data. In this experiment, we examine the relative update transaction (involving search, insert & delete operations) time performance of the three physical model implementations (in Figure 5) over small to large datasets. The results for: (a) small- to medium-sized data (i.e., $23K$ to $23M$ profile pairs), and (b) medium- to large-sized data (i.e., $23M$ to $468M$ profile pairs) are presented in Figure 4 (a) & (b) respectively. The x-axes show the number of profile pairs updated; and the y-axes give the average time, in seconds (on a log2 scale), taken to perform update transactions (over five iterations).

For case (a) above, the HBase-1 and HBase-2 implementations are on a single-node cluster for a fair comparison with the Postgres implementation; and for case (b), the HBase implementations are on a two-node cluster. The results show that in all cases, of the three implementation options, the relational option (Postgres) is significantly slower than the HBase counterparts; and the HBase-1

[4]It is noteworthy that the performance difference of the Postgres implementation for ON/OFF AUTOCOMMIT settings is marginal. Thus, we report the best (i.e., AUTOCOMMIT OFF).

implementation (i.e., option (b) in Figure 5) is the better of the two HBase options. It is also noteworthy that there is no significant performance difference between the HBase implementations on single-node and two-node clusters.

Stress test of HBase implementations. In this experiment, we perform further tests to examine the insertion and update (replacement) operations of the best-performing models (i.e., the two HBase models). We consider three data sizes: 6, 30, and 54 billion profile pairs. As the results in Figure 4(c) show, the insertion operations are, as expected, more efficient than update operations for both models over the three datasets. Moreover, both the insertion and update operations are scalable for both implementations on very small-sized (i.e., just a two-node) cluster.

7 CONCLUSION

In this paper, we present the details of the entity linking system that powers our entity linking method called Certus. We describe the architecture of the system, the graph and data models, and index structures used to support the multiplicity and provenance of attribute and relation values, for effective entity linking and resolution. Further, we give the details of the physical model for storing the entity profiles graph, and discuss three different implementations of the structure for storing the similarity-edges. Due to the frequency of the update transaction of similarity-edges, we perform experiments to evaluate the time performance of accessing the similarity structure on two state-of-the-art database management systems (HBase and Postgres) to demonstrate the relative performances of the three different implementations. The empirical results show a generally good performance for all implementation options. In particular, the HBase implementation options, with even just one- or two-node clusters, scale very well for huge data sizes.

ACKNOWLEDGMENTS

This work is supported by Data to Decisions CRC, ILE/ELR Project DC160031, funded by the Australian Commonwealth Government's CRC Programme. And, partly supported by Data61 - CSIRO.

REFERENCES

[1] [n. d.]. Michael Jordan (disambiguation). https://en.wikipedia.org/wiki/Michael_Jordan_(disambiguation). Last Accessed: 2019-07-24.

[2] David Guy Brizan and Abdullah Uz Tansel. 2006. A Survey of Entity Resolution and Record Linkage Methodologies. *Communications of the IIMA* 6, 3 (2006).

[3] Diego Ceccarelli, Claudio Lucchese, Raffaele Perego, Salvatore Orlando, and Salvatore Trani. 2013. Dexter: an Open Source Framework for Entity Linking. *Intl. workshop on Exploiting semantic annotations in information retrieval (ESAIR)* (2013).

[4] Peter Christen. 2012. A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. *TKDE* 24, 9 (2012), 1537 – 1555.

[5] Xiang Dai. 2018. Recognizing Complex Entity Mentions: A Review and Future Directions. *Student Research Workshop, Association for Computational Linguistics (ACL)* (2018).

[6] Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). *CIKM* (2010).

[7] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Annual Meeting of the Association for Computational Linguistics (ACL)* (2005), 363–370.

[8] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2015. On the Reproducibility of the TAGME Entity Linking System. *European Conf. on Information Retrieval* (2015), 436–449.

[9] Hanna Kopcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *VLDB Endowment* 3, 1 (2010).

[10] Nick Koudas, Sunita Sarawagi, and Divesh Srivastava. 2006. Record linkage: similarity measures and algorithms. *SIGMOD Conf.* (2006).

[11] Selasi Kwashie, Jixue Liu, Jiuyong Li, Lin Liu, Markus Stumptner, and Lujing Yang. 2019. Certus: An Effective Entity Resolution Approach with Graph Differential Dependencies (GDDs). *PVLDB* 12, 6 (2019), 653–666.

[12] Phong Le and Ivan Titov. 2018. Improving Entity Linking by Modeling Latent Relations between Mentions. *Annual Meeting Asso. for Computational Linguistics (ACL)* (2018).

[13] David Milne and Ian H. Witten. 2008. Learning to link with Wikipedia. *CIKM* (2008).

[14] Felix Naumann and Melanie Herschel. 2010. *An Introduction to Duplicate Detection.* Morgan & Claypool Publishers. https://doi.org/10.2200/S00262ED1V01Y201003DTM003

[15] Sarah Oppold and Melanie Herschel. 2018. Provenance for Entity Resolution. In *Provenance and Annotation of Data and Processes - 7th International Provenance and Annotation Workshop, IPAW 2018, London, UK, July 9-10, 2018, Proceedings.* 226–230. https://doi.org/10.1007/978-3-319-98379-0_25

[16] Harinder Pal and Mausam. 2016. Demonyms and Compound Relational Nouns in Nominal Open IE. *Workshop on Automated Knowledge Base Construction (AKBC) at NAACL* (2016).

[17] George Papadakis, Jonathan Svirsky, Avigdor Gal, and Themis Palpanas. 2016. Comparative Analysis of Approximate Blocking Techniques for Entity Resolution. *VLDB Endowment* 9, 9 (2016).

[18] Lawrence Philips. 2000. The Double Metaphone Search Algorithm. *C/C++ Users J.* 18, 6 (June 2000), 38–43. http://dl.acm.org/citation.cfm?id=349124.349132

[19] Francesco Piccinno and Paolo Ferragina. 2014. From TagME to WAT: a new entity annotator. *intl workshop on Entity recognition and disambiguation* (2014).

[20] Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 27, 2 (2015), 443–460.

[21] Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)* (2000), 63–70.

[22] Ricardo Usbeck, Michael RÃČÂűder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both Unister, Martin BrÃČÂijmmer, Diego Ceccarelli, Marco Cornolti, and at. al. 2015. GERBIL: General Entity Annotator Benchmarking Framework. *WWW Conf.* (2015).

[23] Qing Wang, Klaus-Dieter Schewe, and Woods Wang. 2015. Provenance-Aware Entity Resolution: Leveraging Provenance to Improve Quality. In *Database Systems for Advanced Applications - 20th International Conference, DASFAA 2015, Hanoi, Vietnam, April 20-23, 2015, Proceedings, Part I.* 474–490. https://doi.org/10.1007/978-3-319-18120-2_28