Semantic-aware blocking for entity resolution

Semantic-Aware Blocking for Entity Resolution

Qing Wang, Mingyuan Cui and Huizhi Liang

Abstract—In this paper we propose a semantic-aware blocking framework for *entity resolution* (ER). The proposed framework is built using *locality-sensitive hashing* (LSH) techniques, which efficiently unifies both textual and semantic features into an ER blocking process. In order to understand how similarity metrics may affect the effectiveness of ER blocking, we study the robustness of similarity metrics and their properties in terms of LSH families. Then we present how the semantic similarity of records can be captured, measured, and integrated with LSH techniques over multiple similarity spaces. In doing so, the proposed framework can support efficient similarity searches on records in both textual and semantic similarity spaces, yielding ER blocking with improved quality. We have evaluated the proposed framework over two real-world data sets, and compared it with the state-of-the-art blocking techniques. Our experimental study shows that the combination of semantic similarity and textual similarity can considerably improve the quality of blocking. Furthermore, due to the probabilistic nature of LSH, this semantic-aware blocking framework enables us to build fast and reliable blocking for performing entity resolution tasks in a large-scale data environment.

Index Terms—Data matching, entity resolution, record linkage, deduplication, blocking, indexing, locality-sensitive hashing, semantic features, semantic similarity, semantic hashing, taxonomy tree

1 Introduction

Imagine that, given a very large collection of publication records from one or more data sets, such as Scopus[1] and PubMed[2], how can we find records that actually refer to the same publication? To answer questions like this, we need to use entity resolution techniques. *Entity resolution* (ER) is the process of identifying records that represent the same real-world entity in one or more data sets. It is a well-known problem that has been extensively investigated in the past decades [11], [18], [22], [32].

Nevertheless, performing ER tasks over large data sets is still computationally challenging. This is largely due to the quadratic complexity $O(n^2)$ of pairwise comparisons, i.e., each record needs to be compared with all others, for the total number n of records. A common solution for this problem is to use *blocking* [11], which groups records into a set of possibly overlapping but small blocks, and only records that potentially represent the same entity are placed into the same block. Blocking leads to a time complexity $O(m^2 \times |B|)$ for the maximal size m of blocks and the number $|B|$ of blocks in the worst case. In doing so, a relatively small subset of record pairs within the same block can be efficiently identified for comparison, and the performance of ER tasks can be drastically improved, e.g., at least four orders of magnitude faster in the case where $n = 10^6$, $m = 10^2$ and $|B| = 10^4$. However, on the other side of the coin, blocking may deteriorate the quality of ER tasks if a significant number of records in different blocks indeed represent the same entity. In a nutshell, the desired criteria for blocking techniques are two-fold:

(1) How to generate blocks efficiently? This requires us to consider both time and space efficiency when constructing blocks, and trade-offs between these two.
(2) How to generate blocks of good quality? This requires us to consider how blocks support ER tasks to be performed efficiently and accurately, and trade-offs between these two.

A number of blocking techniques have previously been studied [12], [14], [22], [32], [35], [36], [39]. In general the goal is to develop techniques that generate blocks such that (i) all comparisons between records within a block will have a certain minimum similarity with each other, and (ii) the similarity between records in different blocks is below this minimum similarity [12]. Nonetheless, many of these techniques have the following limitations: (a) They are only applicable to specific data sets, e.g., the standard blocking method [18] uses blocking keys to group records, which cannot handle records that are highly similar but have different keys, such as "Qing Wang" and "Wang Qing"; (b) They often still need to compute pairwise comparisons for generating blocks, which is computational expensive, e.g., canopy clustering [32] has a time complexity $O(n^2)$ and is inefficient for large data sets; (c) They are merely based on textual similarity of records, while semantic information is ignored. When a data set is dirty, i.e., containing a significant amount of missing and erroneous data, blocking based on textual similarity often yields poor results [11], e.g., two publication records may have the exactly same title but are semantically different because one is a conference article and the other is a technical report.

In this paper, we propose a semantic-aware LSH blocking framework that can circumvent the above limitations. *Locality-sensitive hashing* (LSH) is a popular technique used for approximately finding nearest neighbors in a high dimensional space [20], [23]. The central idea is to ensure that

- *Qing Wang and Mingyuan Cui are with the Research School of Computer Science, College of Engineering and Computer Science, The Australian National University, Canberra ACT 0200, Australia. E-mail: {qing.wang, mingyuan.cui}@anu.edu.au*
- *Huizhi Liang is with the Department of Computing and Information Systems, The University of Melbourne, Australia. E-mail: huizhi.liang@unimelb.edu.au*

Her research was funded by the Australian Research Council (ARC), Veda Advantage, and Funnelback Pty. Ltd., under Linkage Project LP100200079.

1. http://www.scopus.com/home.url
2. http://www.ncbi.nlm.nih.gov/pubmed/

REC	TITLE	AUTHORS	PUBLISHER
r_1	The cascade-correlation learning architecture	E. Fahlman and C. Lebiere	NISPS Proceedings
r_2	Cascade correlation learning architecture	E. Fahlman & C. Lebiere	Neural Information Systems
r_3	A genetic cascade correlation learning algorithm		Proceedings on Neural Ntw.
r_4	The cascade corelation learning architecture	Fahlman, S., & Lebiere, C.	TR
r_5	Controlled growth of cascade correlation nets		Technical Report (TR)
r_6	The cascade-correlation learn architecture	Lebiere, C. and Fahlman, S.	

	r_1, r_2	r_1, r_3	r_1, r_4	r_1, r_5	r_1, r_6	r_2, r_3	r_2, r_4	r_2, r_5	r_2, r_6	r_3, r_4	r_3, r_5	r_3, r_6	r_4, r_5	r_4, r_6	r_5, r_6
TS	0.88	0.70	0.90	0.40	0.85	0.60	0.80	0.40	0.85	0.70	0.40	0.60	0.40	0.85	0.40
SS	0.50	1.00	0.00	0.00	0.17	0.50	0.00	0.00	0.17	0.00	0.00	0.17	0.17	0.17	0.17

B_1 : r_1, r_2, r_4, r_6 | r_3 | r_5 $\quad$ B_2 : r_1, r_2, r_3, r_6 | r_4, r_5, r_6 $\quad$ B_3 : r_1, r_2, r_6 | r_4, r_6 | r_3 | r_5

Fig. 1. Blocking with textual similarity (TS) and semantic similarity (SS): B_1 based on textual similarity; B_2 based on semantic similarity; B_3 based on both textual similarity and semantic similarity

the more similar two points are, the higher the probability is that they are hashed into the same bucket. The reasons why we use LSH for blocking are based on two observations. First, blocking is intimately related to the nearest neighbor search problem in similarity spaces [3]. Existing ER techniques [11] are largely grounded on the assumption that the more similar records are, the more likely they represent the same entity. Although this observation is not universally valid, it generally holds in practice [4]. Therefore, LSH is well suited for searching similar records that may represent the same entity with certain probability. Second, LSH readily lends itself to fast blocking techniques over very large data sets due to its probabilistic nature. The time complexity of generating blocks can be decreased to $O(n)$, which provides attractive scalability potential. In addition to time efficiency, we also observe that leveraging LSH techniques over multiple similarity spaces (e.g., textual similarity and semantic similarity) may considerably improve the quality of blocking, i.e., reducing redundant or unnecessary pairs in blocks without loss of accuracy.

Example 1.1. *Suppose that r_1, r_2 and r_3 in Fig. 1 are conference articles, r_4 and r_5 are technical reports, and r_6 is semantically ambiguous (e.g., could be a conference article, a technical report or others). Based on the textual similarity of titles and authors, $\{r_1, r_2, r_4, r_6\}$ can be placed into the same block, which leads to the set B_1 of blocks. Based on their semantic similarity, we may have the set B_2 of blocks. In either case, the quality of blocks is not satisfactory because neither r_4 nor r_3 should be in the same block with r_1 and r_2. However, if we consider both textual and semantic similarities, we would have the set B_3 of blocks with improved quality. From Fig. 1, we can see that B_3 only has 4 pairs of records to be compared for resolving these records, whereas B_1 and B_2 require to compare 6 and 9 pairs of records, respectively.*

Contributions. We develop a semantic-aware blocking framework for ER using LSH techniques, and show that the robustness of similarity metrics plays a critical role in handling the blocking problem for ER. In particular, the robustness of similarity metrics serves as a bridge between the blocking problem and the nearest neighbor search problem. In principle, the blocking problem and the nearest neighbor search problem have different concerns. The former is concerned with approximately grouping records in accordance with how they refer to real-world entities, whereas the latter is concerned with approximately finding similar records. Since records that are close to each other in one similarity space do not necessarily refer to the same real-world entity, the effectiveness of searching nearest neighbors for blocking relies on the robustness of the chosen similarity metrics.

We then explore semantic similarity for improving the quality of blocking. There were several challenges in our work: (1) how to find useful semantic information from missing, inconsistent and noisy data; (2) how to develop a metric for quantifying the semantic similarity among records for the purpose of blocking; (3) how to incorporate semantic similarity with textual similarity. We address these challenges by detecting missing and inconsistent data patterns, exploiting relationships between records and semantic concepts in terms of taxonomy trees from domain knowledge, and building semantic hashing families. In doing so, we integrate both textual and semantic similarity features into a unified LSH blocking method for ER.

We have evaluated our framework over two real-world data sets. The experimental results show that integrating semantic features and textual features into the blocking process can significantly improve the quality of blocks, particularly when data sets are imperfect (i.e., contain inaccurate, incomplete or erroneous data). In such cases, the sizes of blocks generally become smaller because semantic features can effectively eliminate record pairs that are textually similar but semantically dissimilar, which often represent different entities in real-world applications. We have also compared our framework with 12 state-of-the-art blocking techniques, and it turns out that the semantic-aware framework has the best blocking quality (i.e., the highest FM values) over both data sets.

In this paper we are only concerned with blocking techniques. Nonetheless, our blocking results can be used as input to any ER algorithms for classifying records [11].

2 Related Work

Studies on ER have been heavily carried out over the last 50 years [11], [18]. In practice, ER tasks are commonly performed in two stages: (1) blocking – group records that might represent the same entity into the same block; (2) clustering – classify records into clusters such that each cluster represents a distinct entity. In contrast to clustering, which

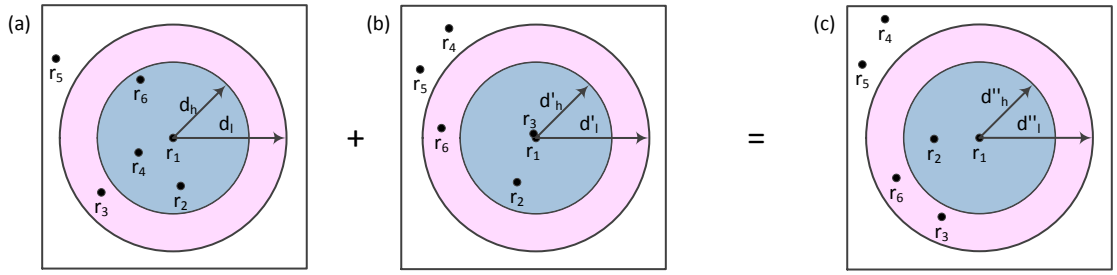


Fig. 2. (a) Textual similarity space; (b) Semantic similarity space; (c) Combining similarity spaces (high, low and uncertain regions are highlighted by dark, white and light colors, respectively)

is in search for exact solutions and often computationally costly (i.e., trading-off efficiency for certainty and precision), blocking aims to be approximate and computationally cheap (i.e., trading-off certainty and precision for efficiency). Previously, blocking has been studied by many works [7], [12], [14], [22], [26], [27], [32], [39]. These existing blocking techniques nonetheless have some limitations as we have discussed before, including: generating blocks inefficiently over large data sets [14], [32], or filtering out true matches that are textually dissimilar but semantically similar during the blocking process [7], [12], [22], [26], [27], [39].

LSH was originally introduced for solving approximate nearest neighbor search problem [20], [23]. Some variants of LSH [5], [29], [34] have been proposed to improve the quality of the original LSH, which offer different trade-offs between time and space complexity. The entropy-based LSH [34] and multi-probe LSH [29] both aimed to reduce the number of hash tables required by the original LSH while achieving the same accuracy. The LSH forest [5] represents each hash table by a prefix tree and the number of hash functions per hash table can be tuned in terms of different distance metrics. In the context of ER, LSH has been used in several works [24], [31], [39], e.g., HARRA [28] was proposed as an iterative LSH-based ER method. All these works are based on textual similarity, but our work considers blocking in terms of not only textual similarity but also semantic similarity. To detect near duplicate RDF resources, a LSH-based approach was introduced in [24], which measures semantic similarity based on the textual representation of RDF metadata. Our work is different as we measure semantic similarity in terms of the structural relatedness of concepts in taxonomy trees and their is no need to compare the textual similarity of these concepts.

Semantic technologies have important influences in a variety of areas, such as improving search on the web and semantic indexing in information retrieval [19], [33]. Taxonomical knowledge can be modelled using ontologies, which have been extensively studied in the past, and used in a wide range of applications [30], [38]. Previous studies focused primarily on measuring semantic relatedness using path-length and information content approaches [38]. Unlike them, we are only interested in measuring semantic similarity of records in terms of their likeliness of representing the same entity. Therefore, we define semantic similarity of records with respect to their semantic concepts satisfying certain properties in a given taxonomy tree. This allows us to generate binary signatures for LSH blocking and preserve semantic similarity among records. To our best of knowledge, little work has been done on leveraging semantic similarity for blocking.

3 Problem Definition

Let R be a finite, non-empty set of records, each having a finite number of attributes. In viewing that records are a set of data points, a *distance space* (R, δ) can be defined for records in R together with a *distance metric* $\delta : R \times R \mapsto [0, 1]$ that satisfies the non-negativity, identity, symmetry and triangle inequality properties as defined in [42].

Each distance metric δ corresponds to a similarity metric sim such that $\delta(x, y) = 1 - sim(x, y)$, and (R, sim) is a *similarity space*. A similarity metric sim is *γ-robust* over R where $\gamma \in [0, 1]$ if, for any two record pairs whose similarity difference is greater than $1-\gamma$, the record pair with a higher similarity value is more likely to represent the same entity than the other pair. Let E be a set of entities, $e : R \mapsto E$ be a function that maps each record $r \in R$ to an entity $e(r) \in E$ which it represents, and $Pr[e(r_1) = e(r'_1)]$ be the probability of $e(r_1) = e(r'_1)$. Formally, the notion of *γ-robust similarity metric* sim is defined as:

$$Pr[e(r_1) = e(r'_1)] \geq Pr[e(r_2) = e(r'_2)] \text{ if } \quad (1)$$
$$sim(r_1, r'_1) - sim(r_2, r'_2) \geq 1 - \gamma.$$

Thus, the probability that two records represent the same entity positively correlates with the similarity of the records if the similarity increases or decreases more than $1 - \gamma$. The greater γ is, the more robust a similarity metric sim can be used for the ER blocking purpose.

Given a record $r \in R$, the other records $r' \in R$ whose distances with r are at most k, i.e., $\delta(r, r') \leq k$, are called the *k-distance neighbors* of r in the distance space (R, δ). In accordance with Equation 1, for each record $r \in R$, the distance space (R, δ) is divided into three regions by two distance values d_l and d_h with $d_l - d_h \geq \gamma$: (1) *high region*: records whose distances with r are not greater than d_h have a high probability of representing the same entity as r; (2) *low region*: records whose distances with r are greater than d_l have a low probability of representing the same entity as r; (3) *uncertain region*: records whose distances with r are between d_h and d_l are ambiguous in the sense that the probability of representing the same entity as r is uncertain.

The notion of k-distance neighbors can be further generalized to multiple similarity spaces. Let R be a set of records, and $[(R, \delta_1), \ldots, (R, \delta_n)]$, abbreviated as $(R, \vec{\delta})$ for $\vec{\delta} = [\delta_1, \ldots, \delta_n]$, be a number of distance spaces that R associates with. Then given a record $r \in R$ and a parameter

vector $\vec{k} = [k_1, \ldots, k_n]$ where $k_i \in [0, 1]$ $(i = 1, \ldots, n)$, all the records $r' \in R$ whose distances with r are at most $\vec{k}$ (i.e., $\delta_i(r, r') \leq k_i$ in (R, δ_i) for every i) are called the *$\vec{k}$-distance neighbors* of r in multiple distance spaces $(R, \vec{\delta})$.

Example 3.1. *Fig. 2.(a)-(b) depicts the textual and semantic spaces for the records in Fig. 1, respectively. Both r_2 and r_4 have a high probability of being textually similar to r_1 in Fig. 2.(a). However, in terms of semantic similarity with r_1 in Fig. 2.(b), r_2 retains a high probability while r_4 has a low probability. When considering both similarity spaces, as shown in Fig. 2.(c), we know that r_2 likely represents the same entity as r_1 but it is unlikely for r_4 and r_5 despite that r_4 and r_1 are textually similar with r_1.* □

Let $(R, \vec{\delta})$ be multiple similarity spaces. Then *the blocking problem* over $(R, \vec{\delta})$ is to determine a parameter vector $\vec{d_l}$ such that a set B of blocks is generated, in which only $\vec{d_l}$-distance neighbors of a record r in R are placed into the blocks that contain r. Intuitively, $\vec{d_l}$ refers to the distance values between low and uncertain regions in $(R, \vec{\delta})$. The *optimization version* of the blocking problem can be stated in terms of the set P of all *true matches* (i.e., record pairs that represent the same entity) and the set N of all *true non-matches* (i.e., record pairs that represent two different entities) with the objective as:

$$argmin_{\theta_B} \quad \frac{\sum_{(r_1,r_2)\in N} \theta_B(r_1, r_2)}{\sum_{r_1 \neq r_2, r_1 \in R, r_2 \in R} \theta_B(r_1, r_2)} \tag{2}$$

such that

$$1 - \frac{\sum_{(r_1,r_2)\in P} \theta_B(r_1, r_2)}{|P|} \leq \epsilon,$$

where $|P| + |N| = |R|^2$, θ_B is a blocking function that takes two records as input, and returns 1 if there is at least one block of B containing both records and 0 otherwise, and ϵ is an error ratio indicating the percentage of true matches that are lost in blocks B.

4 Semantic Similarity

In this section we discuss semantic similarity between records, i.e., how much two records are alike semantically.

4.1 Taxonomy Trees

In real-world applications, domain knowledge commonly exists. It may be acquired from domain experts or from existing ontologies, corpus, or thesaurus resources in a knowledge base, such as Wikipedia [19] and WordNet [33]. Records in the ER process are often related to semantic concepts in such knowledge bases. Thus, a collection of taxonomy trees, which describe how the semantic concepts of records are related, can be constructed either manually, or derived from knowledge bases. We first define the notions of concept and taxonomy tree [16], [40].

Definition 4.1. *A* (semantic) concept *is an abstract set of things. A* taxonomy tree *t consists of a set C_t of nodes, each representing a concept, and a set of edges that represent a subsumption relation between concepts in C_t.*

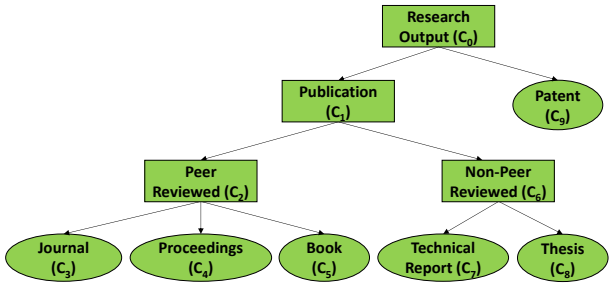


Fig. 3. A bibliographic taxonomy tree t_{bib}

Conceptually, a partial order $\preceq$ exists between concepts in a taxonomy tree, i.e., $c_1 \preceq c_2$ means that the concept c_1 is subsumed by the concept c_2. For each taxonomy tree, from the root to a leaf, the concepts vary from being most general to being more specific. We use $child(c)$ to denote the set of concepts represented by the child nodes of c, and any two concepts in $child(c)$ are not subsumed one by the other.

Example 4.1. *Fig. 3 shows a taxonomy tree t_{bib} from the bibliographic domain, which has concepts in a hierarchical structure in terms of the subsumption relation. Each node represents a semantic concept in the bibliographic domain such as journal, book and technical report, and each edge represents a subsumption relationship between two semantic concepts, e.g., $c_3 \preceq c_1$, $c_4 \preceq c_1$ and $c_5 \preceq c_1$ because journals, conference proceedings and books are considered as different types of publications.* □

4.2 Semantic Analysis

Due to various reasons, such as incomplete data or errors, the semantics of a record may be ambiguous. In such cases, the semantic interpretation of a record is a number of possible concepts. Formally, we define the semantic interpretation of records w.r.t. one or more taxonomy trees by a semantic function.

Definition 4.2. *Let R be a set of records, T be a set of taxonomy trees with the set $C_T = \bigcup_{t\in T} C_t$ of nodes, and $\mathcal{P}(C_T)$ be the powerset of C_T. A* semantic function *$\zeta : R \mapsto \mathcal{P}(C_T)$ assigns each $r \in R$ with a subset of concepts in $\mathcal{P}(C_T)$ as its semantic interpretation s.t. the following properties are satisfied:*

(a) Specificity*: $\zeta(r)$ contains a set of concepts that are as specific as possible, i.e., $\forall c, c' \in \zeta(r). c \preceq c' \Rightarrow c = c'$;*
(b) Isolation*: $\zeta(r)$ generates a set of concepts that are related to r without accessing any records in $R - \{r\}$.*

Intuitively, the interpretation $\zeta(r)$ of each record r is a set of concepts in taxonomy trees. In accordance with the subsumption relationship represented by edges, if a concept c is in $\zeta(r)$, then any other concept that subsumes c cannot coexist in $\zeta(r)$. Thus, by the specificity property, only the most specific concept remains in the interpretation. The purpose of the isolation property is to ensure the efficiency of a semantic function.

There are a variety of ways to define the semantic function for records in terms of taxonomy trees, such as mining patterns, using meta-data, and leveraging relationships. The following example illustrates that a semantic function can be simply defined by analyzing values of an attribute.

Example 4.2. *For the records r_1-r_6 shown in Fig. 1, we may analyze the values in the attribute* PUBLISHER *in terms of the taxonomy tree depicted in Fig. 3, and obtain the following semantic interpretation: $\zeta(r_1) = \{c_4\}$, $\zeta(r_2) = \{c_2\}$, $\zeta(r_3) = \{c_4\}$, $\zeta(r_4) = \{c_7\}$, $\zeta(r_5) = \{c_7\}$ and $\zeta(r_6) = \{c_0\}$.* □

4.3 Similarity Metric

As previously studied [38], many approaches have been proposed to measure the semantic relatedness of records. For simplicity, we use a simple approach to derive the semantic similarity of two records based on the semantic similarity of their related concepts in taxonomy trees. Nevertheless, other more sophisticated approaches may also be used to measure semantic similarity within this framework.

Similarity between concepts. Given a set T of taxonomy trees, each concept c in T corresponds to a subtree rooted at c, which is denoted as $t(c)$. Let $leaf(c)$ be the set of concepts represented by all leaves in $t(c)$, and $sim_S(c_1, c_2)$ denote the semantic similarity of two concepts c_1 and c_2. One of the important properties which we want to ensure for semantic similarity between two concepts is as follows:

$$\forall c_1, c_2 \in child(c)\ c_1 \neq c_2 \Rightarrow sim_S(c_1, c_2) = 0. \quad (3)$$

Example 4.3. *Consider the taxonomy tree depicted in Fig. 3. $sim_S(c_3, c_5)$ should be zero since c_3 represents journal articles and c_5 represents books.* □

Thus, in viewing that each concept can be alternatively represented by a set of concepts in its leaves, we measure the semantic similarity of two concepts c_1 and c_2 in accordance with their leaves in $t(c_1)$ and $t(c_2)$, which is similar to Jaccard similarity coefficient [11], i.e.,

$$sim_S(c_1, c_2) = \frac{|leaf(c_1) \cap leaf(c_2)|}{|leaf(c_1) \cup leaf(c_2)|}. \quad (4)$$

This metric takes into account the subsumption relationship among concepts. For three concepts c_1, c_2 and c_3 satisfying $c_3 \preceq c_2 \preceq c_1$, we always have $sim_S(c_1, c_3) \leq sim_S(c_2, c_3)$ and $sim_S(c_1, c_3) \leq sim_S(c_1, c_2)$.

Example 4.4. *Consider the taxonomy tree depicted in Fig. 3 again. We have $sim_S(c_0, c_1) = 5/6$ because the intersection and union of $leaf(c_0)$ and $leaf(c_1)$ are 5 and 6, respectively. Similarly, we can obtain $sim_S(c_1, c_2) = 3/5$, $sim_S(c_0, c_4) = 1/6$ and $sim_S(c_2, c_6) = 0$.* □

Similarity between records. Semantic similarity between records is measured on the basis of their related concepts. More specifically, given two records r_1 and r_2, the semantic similarity of r_1 and r_2 is defined as

$$sim_S(r_1, r_2) = \sum_{(c_1,c_2) \in P(r_1,r_2)} \frac{|\alpha(c_1, c_2)|}{|\beta(r_1, r_2)|} \cdot sim_S(c_1, c_2), \quad (5)$$

where $P(r_1, r_2) = \{(c_1, c_2) | c_1 \in \zeta(r_1), c_2 \in \zeta(r_2),$ and $c_1 \preceq c_2 \vee c_2 \preceq c_1$holds$\}$ is a set of related concept pairs between r_1 and r_2, $\alpha(c_1, c_2) = leaf(c_1) \cup leaf(c_2)$ and $\beta(r_1, r_2) = \bigcup_{c_1 \in \zeta(r_1), c_2 \in \zeta(r_2)} \alpha(c_1, c_2)$. Intuitively, $\frac{|\alpha(c_1,c_2)|}{|\beta(r_1,r_2)|}$ is the weight of (c_1, c_2), which indicates the influence of the semantic similarity from each pair (c_1, c_2) of related concepts on the semantic similarity of (r_1, r_2). When two records r_1 and r_2 are both interpreted to exactly the same concept in a taxonomy tree, i.e., $\zeta(r_1) = \{c_1\}$ and $\zeta(r_2) = \{c_2\}$, the semantic similarity between the records coincides with the semantic similarity between their related concepts, i.e., $sim(r_1, r_2) = sim(c_1, c_2)$ holds.

We thus have the following two propositions. Proposition 4.1 states that if a record r is interpreted to all concepts represented by the child nodes of a concept c, then it is equal to being interpreted as c directly. Proposition 4.2 states that the semantic similarity of two records is 0 iff their concepts are not related, i.e., there is no path between their concepts.

Proposition 4.1. *If $\zeta(r_1) = \{c\}$ and $\zeta(r_2) = child(c)$, then $sim_S(r_1, r_2) = 1$.*

Proposition 4.2. *$sim_S(r_1, r_2) = 0$ iff $P(r_1, r_2) = \emptyset$.*

Example 4.5. *Based on the semantic interpretation of the records r_1-r_6 in Section 4.2, and the taxonomy tree depicted in Fig. 3, we have $sim_S(r_1, r_2) = 1/2$ because $\zeta(r_1) = \{c_4\}$, $\zeta(r_2) = \{c_3, c_4\}$, $P(r_1, r_2) = \{(c_4, c_3), (c_4, c_3)\}$, $sim_S(c_4, c_3) = 0$, $sim_S(c_4, c_4) = 1$ and thus $sim_S(r_1, r_2) = (\frac{2}{2} \cdot 0) + (\frac{1}{2} \cdot 1) = 1/2$. Similarly, we can obtain $sim_S(r_3, r_2) = 1/2$, $sim_S(r_1, r_3) = 1$, $sim_S(r_1, r_5) = 0$, $sim_S(r_2, r_6) = 1/3$ and $sim_S(r_1, r_6) = sim_S(r_5, r_6) = 1/6$.* □

4.4 Semantic Hashing

In order to efficiently analyze the semantic similarity of records, we need to convert the semantic interpretation of each record (i.e., a set of semantic concepts in taxonomy trees) into a binary vector, called *semhash signature*, and such semhash signatures should approximately preserve the semantic similarity between records. For this, we develop *semhash functions* in the following.

Given a set R of records that are interpreted in terms of taxonomy trees T and $C_R = \sum_{r \in R} \zeta(r)$, a family of semhash functions is defined in one-to-one correspondence with a subset $C \subseteq C_T$ of concepts (i.e., semantic features of interest) in T such that the following conditions are satisfied:

(1) *Disjointness*: $\forall c_1, c_2 \in C\ (c_1 \not\preceq c_2 \wedge c_2 \not\preceq c_1)$ (concepts in C are pairwise disjoint);
(2) *Completeness*: $\forall c_1 \in C_R\ (leaf(c_1) \subseteq C)$ (all concepts that are related to records in R are in C);
(3) *Non-emptiness*: $\forall c_1 \in C \exists c_2 \in C_R\ (c_1 \preceq c_2)$ (each concept in C is related to at least one record in R).

Each concept $c_i \in C$ corresponds to a semhash function g_i that takes a record r as input and produces 0 or 1 as output, where 1 (resp. 0) indicates that r is related (resp. not related) to c_i. In doing so, we obtain the semantic signature $G(r) = [g_1(r), g_2(r), \ldots, g_n(r)]$ for each record $r \in R$ after applying the semhash functions $G = \{g_1, \ldots, g_n\}$, and a semantic signature matrix for all records. Algorithm 1 describes the details of generating semhash signatures, which takes time $O(|R| + |C_T|)$ in the worse case.

Input: a set R of records and a set T of taxonomy trees
Output: a semantic signature matrix M for R

(1) Select a subset C of concepts in T for semhash functions $G = \{g_1, \ldots, g_{|C|}\}$;
(1.1) For each $c \in \sum_{r \in R} \zeta(r)$, add $leaf(c)$ into C;
(2) For each $r \in R$, check the concepts in $\zeta(r)$ w.r.t. c_i $(i = 1, \ldots, |C|)$ and do the following:

$$g_i(r) = \begin{cases} 1 & \text{if } \exists c \in \zeta(r).c_i \preceq c \text{ holds;} \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 1: Generating semhash signatures

In accordance with the definition of semhash function, we have the following proposition.

Proposition 4.3. *Let $G = \{g_1, \ldots, g_n\}$ be the set of semhash functions chosen for R, and $sim_J(v_1, v_2)$ be the Jaccard similarity coefficient between two vectors v_1 and v_2 of the same length. Then for any $r_1, r_1', r_2, r_2' \in R$, we have $sim_J(G(r_1), G(r_1')) \geq sim_J(G(r_2), G(r_2'))$ iff $sim_S(r_1, r_1') \geq sim_S(r_2, r_2')$.*

Note that, it is possible to combine semhash and minhash functions [8] for generating semantic signatures. This depends on how many semantic features are considered. In practice, the number of semantic features is often relatively small. Hence, we can use semhash functions to generate semantic signatures directly.

5 Semantic-Aware LSH Blocking

In this section we propose a semantic-aware LSH blocking framework, which incorporates semantic features into the existing LSH techniques for ER blocking. Let $d_1 < d_2$ be two distance values of a distance metric δ, and p_1 and p_2 be two probabilities. Then a family H of functions is (d_1, d_2, p_1, p_2)-*sensitive* over (R, δ) if, for any $r_1, r_2 \in R$ and any $h \in H$, the following conditions are satisfied: (1) if $\delta(r_1, r_2) \leq d_1$, then $Pr[h(r_1) = h(r_2)] \geq p_1$; (2) if $\delta(r_1, r_2) \geq d_2$, then $Pr[h(r_1) = h(r_2)] \leq p_2$, where $Pr[h(r_1) = h(r_2)]$ is the probability of $h(r_1) = h(r_2)$ [23]. Although a LSH family can be established for any distance space (R, δ) [9], two problems still remain:

(1) How does the robustness of a similarity metric affect the effectiveness of LSH for blocking?
(2) How can LSH families for textual and semantic distance spaces be integrated in an efficient way?

By Equation 1 and the notion of LSH family, we have the following proposition that answers the first question. Intuitively, it states that, for a distance metric that is γ-robust, and two pairs of records from R whose difference between the similarity values is at least γ, the probability of hashing each pair into the same value correlates positively to the likelihood of representing the same entity by the pair.

Proposition 5.1. *Let H be a (d_1, d_2, p_1, p_2)-sensitive family over (R, δ) and δ is γ-robust. Then for any $r_1, r_2, r_1', r_2' \in R$ and each $h \in H$, whenever $\delta(r_1, r_1') \leq d_1$, $\delta(r_2, r_2') \geq d_2$ and $d_2 - d_1 \geq 1 - \gamma$ hold, we have*

$$Pr[e(r_1) = e(r_1')] \geq Pr[e(r_2) = e(r_2')] \text{ if} \quad (6)$$
$$Pr[h(r_1) = h(r_1')] \geq Pr[h(r_2) = h(r_2')].$$

For the second question, we will discuss it in Section 5.2.

5.1 LSH with Minhash Signatures

To measure the textual similarity of two records, a commonly used LSH technique is *minhash functions* [8]. Given a set R of records, there are three main steps.

(1) *Shingling*: convert each record into a binary vector in accordance with some selected attributes A of R. Thus, a set $\{a_1, \ldots a_m\}$ of q-grams [11] from the values of A in all records is first collected, then each record r is represented as a binary vector $[v_1, \ldots, v_m]$ such that if a q-gram a_i $(i \in [1, m])$ occurs in r, then the value of v_i is set to 1; otherwise it is 0.
(2) *Minhashing*: generate a signature vector (i.e., so-called *minhash signature*) for each record using minhash functions. For this, a number of minhash functions $h_1, h_2, \ldots, h_n$ are used, where n is much smaller than m, yielding a signature vector $[h_1(r), h_2(r), \ldots, h_n(r)]$ for each record r. In doing so, records have small signatures that approximately preserve textual similarity between them.
(3) *Amplifying*: control the locality sensitivity by dividing the minhash functions $h_1, h_2, \ldots, h_n$ into a number l of hash tables of equal size k where $l \cdot k = n$. Since records whose minhash signatures agree in one of such hash tables are placed into the same block, this turns a (d_1, d_2, p_1, p_2)-sensitive family into a $(d_1, d_2, 1-(1-p_1^k)^l, 1-(1-p_2^k)^l)$-sensitive family.

The following proposition states that such a LSH family with minhash functions can ensure that two textually identical records can always be hashed into the same block. Meanwhile, two records that are textually dissimilar may still be hashed into the same block although the probability could be low, e.g., "deduplication" and "entity resolution" are textually dissimilar but refer to the same problem.

Proposition 5.2. *Let H_t be a LSH family with minhash functions. Then, for any $r_1, r_2 \in R$ and $h \in H_t$, we have:*

(1) *if $sim_J(r_1, r_2) = 1$, then $Pr[h(r_1) = h(r_2)] = 1$, and*
(2) *if $sim_J(r_1, r_2) = 0$, then $Pr[h(r_1) = h(r_2)] \geq 0$.*

5.2 Integrating Semhash Signatures

How should we develop a LSH blocking method that combines textual and semantic similarity spaces for improving the quality of blocks? Before exploring possible solutions, it is important to note that textual and semantic features are different in at least two aspects.

(1) Textual and semantic features are not equally sensitive for identifying entities: (i) two records of the same entity may have dissimilar textual representation but they are often semantically related with each other, and (ii) two records whose textual representations are highly similar (e.g., records that have the same title) are more likely to refer to the same entity than two records whose semantic interpretations are highly similar (e.g., records that are both journals).
(2) The dimensionality of semantic features is often relatively small (i.e., range from several to hundreds), while the dimensionality of their textual features can be very large, e.g., $17,576$ for 3-grams and $456,976$ for 4-grams if only 26 characters are used.

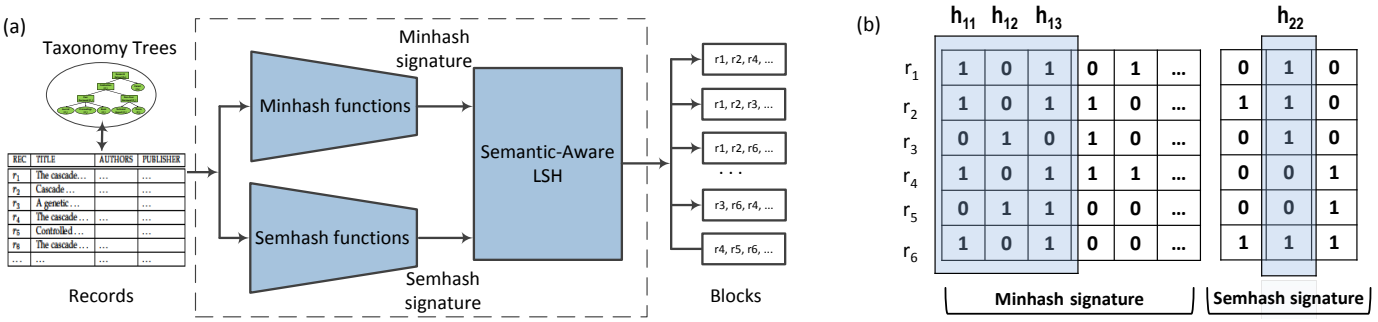


Fig. 4. An illustration of the semantic-aware LSH framework with minhash and semhash signatures

We need to cater for these differences when incorporating textual and semantic similarities into the LSH blocking. Therefore, we propose the semantic-aware LSH blocking by augmenting a LSH family for textual similarity with a *w-way semantic hash function*. The key ideas are as follows. Assume that, for a semhash function $g \in G$, the probability of having the value 1 for a randomly chosen record is p_v, i.e., $p_v = Pr[g(r) = 1]$, and the probability that two randomly chosen records r_1 and r_2 have the same value in g is p_e, i.e., $p_e = Pr[g(r_1) = g(r_2)]$. We may assume $Pr[g(r_1) = g(r_2)] = \sum_{g_i \in G} Pr[g_i(r_1) = g_i(r_2)]/|G|$ if $|G|$ is sufficiently large. A LSH family H_g for semantic similarity is a set of hash functions, each $h_g \in H_g$ being determined by a semhash function g uniformly chosen at random from G such that, h_g yields *true* for two records r_1 and r_2 if both records have the value 1 for g, and yields *false* otherwise. The probability that two records have the *true* value for a hash function $h_g \in F$ is thus $p_v \cdot p_e$. Based on H_g, two types of w-way semantic hash functions can be constructed:

(1) A *w-way AND function* $h_{[w,\wedge]}$ is built upon w randomly chosen functions $\{h_{g1}, \ldots, h_{gw}\}$ from H_g such that $h_{[w,\wedge]}(r_1, r_2) = true$ iff $h_{g1}(r_1, r_2) \wedge \cdots \wedge h_{gw}(r_1, r_2) = true$;

(2) A *w-way OR function* $h_{[w,\vee]}$ is built upon w randomly chosen functions $\{h_{g1}, \ldots, h_{gw}\}$ from H_g such that $h_{[w,\vee]}(r_1, r_2) = true$ iff $h_{g1}(r_1, r_2) \vee \cdots \vee h_{gw}(r_1, r_2) = true$.

Let $s' = p_v \cdot p_e$, l be the number of hash tables, and k be the number of hash functions in each hash table. Then the probability that two records with the textual similarity s and the semantic similarity s' are placed into the same block is $1 - (1 - s^k \cdot p)^l$, where p indicates the probability that the augmented w-way semantic hash function for the two records returns *true*, and is defined as:

$$p = \begin{cases} (s')^w & \text{if } \mu = \wedge; \\ 1 - (1 - s')^w & \text{if } \mu = \vee. \end{cases}$$

The following proposition states that, for any two records that are semantically dissimilar, regardless of their textual similarity, the collision probability of these two records in a semantic-aware LSH family is always 0.

Proposition 5.3. *Let H_{ts} be a semantic-aware LSH family. Then, for any $r_1, r_2 \in R$ and $h \in H_{ts}$, we have:*

(1) *if $sim_S(r_1, r_2) = 0$, then $Pr[h(r_1) = h(r_2)] = 0$, and*
(2) *if $sim_J(r_1, r_2) = 1$, then $Pr[h(r_1) = h(r_2)] \leq 1$.*

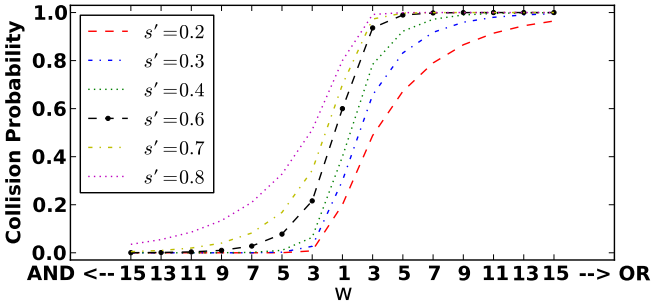


Fig. 5. The collision probability of a w-way semantic hash function under different semantic similarities $s\prime$, where $w = 1, 2, \ldots, 15$ and $\mu \in \{\wedge, \vee\}$

Fig. 4.(a) provides a high-level illustration of the proposed semantic-aware LSH blocking framework. Given a set of records and several taxonomy trees as input, minhash and semhash functions first generate minhash and semhash signatures in terms of the textual and semantic similarities among these records, respectively. Then, a number of blocks are generated by combining the minhash and semhash signatures. The quality of blocks can thus be improved by leveraging both textual and semantic features.

Example 5.1. *Consider our running example with the records r_1-r_6 in Fig. 1 and the taxonomy trees t_{bib} in Fig. 3. Suppose that the minhash and semhash signatures of the records w.r.t. t_{bib} are presented in Fig. 4.(b), where one hash table of the LSH family is constituted by h_{11}, h_{12} and h_{13}, and a 1-way OR semantic function is built upon h_{22}. According to the hash table $[h_{11}, h_{12}, h_{13}]$, the records r_1, r_2, r_4 and r_6 are textually similar, and would be hashed into the same block if only considering textual similarity. However, when taking into account the 1-way OR semantic function, we would have that r_4 is not semantically similar to r_1, r_2 and r_6, and cannot be placed into the same blocks with these records. Nevertheless, r_1, r_2 and r_6 will be hashed into the same block because they are both textually and semantically similar (i.e., their values are the same in $[h_{11}, h_{12}, h_{13}]$ and their values equal to 1 in h_{22}).* □

5.3 Parameter Tuning

The parameter tuning of our semantic-aware blocking method is primarily based on the similarity of records. For this, we first need to select a list $\{A_1, \ldots, A_n\}$ of attributes used for ER blocking and similarity functions f_i such that $f_i(r_1.A_i, r_2.A_i) \in [0, 1]$ where $1 \leq i \leq n$, and $r_1.A_i$ and $r_2.A_i$ refer to the values of A_i in the records r_1 and r_2, respectively. The selection of attributes and their similarity functions is a general requirement for any ER methods and has been well studied in previous works [11], [26].

Thus, below we focus on discussing the parameters that are specific to our method: (1) *the number l of hash tables*, (2) *the size k of hash tables* (i.e., the number of minhash functions per hash table), and (3) *a w-way semantic hash function*.

Generally, there are three steps for tuning the parameters: (i) determine two similarity thresholds s_h and s_l w.r.t. a desired error ratio ϵ, which correspond to the distance values d_h and d_l previously discussed in Section 3 (i.e., $s_h = 1 - d_h$ and $s_l = 1 - d_l$); (ii) determine the parameters l and k based on s_l and s_h, and their desired collision probabilities p_l and p_h; (iii) determine the parameter w based on the quality of semantic features. For Step (i), we first need to learn the probability density function $f_s(x)$ of true matches over textual similarity from the training dataset. Then, by $\int_0^{s_h} f_s(x)d = \epsilon$, s_h can be automatically determined given a desired ϵ. Similarly, s_l may be determined so that records whose textual similarity is lower than s_l are considered to be in different blocks. For Step (ii), the desired probabilities of s_l and s_h need to be decided. That is, the probability of finding records whose textual similarity is greater than s_h is at least p_h and the probability of finding records whose textual similarity is less than s_l is at most p_l. Since the probability of placing two records with a similarity s into one block is $1 - (1 - s^k)^l$, l and k can be automatically determined based on the conditions $l \leq log_{1-s_h^k}(1 - p_h)$ and $l \geq log_{1-s_l^k}(1 - p_l)$. For Step (iii), if the semantic features are noisy, uncertain (i.e., semantic features of some records are missing) or heterogeneous (different records of the same entities may have different semantic features), a w-way OR semantic function is preferred; otherwise, a w-way AND semantic function may be chosen. The purpose of adding a w-way semantic hash function is to filter out true non-matches whose semantic similarities are less than certain degree. Fig. 5 shows that the choice of the parameter w amplifies the collision probability of semantically similar records such that increasing w in a w-way AND function lowers the probability and increasing w in a w-way OR function increases the probability.

Note that, these parameters can be determined based on a small training dataset. Nonetheless, the performance of our method may be affected, which depends on how close the similarity distribution of records in the training dataset is to the similarity distribution of records in an entire dataset.

6 Experiments

We have implemented our semantic-aware LSH blocking framework to investigate the following three questions:

(1) *Blocking parameters*: How effectively the parameters of the semantic-aware LSH blocking can be determined, in relating to the similarity distribution of a training data set, and desired error ratio and collision probabilities?

(2) *Blocking quality*: Can the semantic-aware LSH blocking yield blocks with better quality, in comparison to the LSH blocking only over textual similarity space? What are the effects of the incorporated semantic hash functions on blocking quality under different settings of w and μ? How do the proposed LSH blocking methods perform in comparison with the state-of-the-art blocking techniques?

(3) *Blocking efficiency and scalability*: How efficiently will the semantic-aware LSH blocking be used for ER? Does it support good scalability for constructing blocks?

Our implementation code is written in Java. The experiments were conducted on a server with 128 GBytes of main memory and two 6-core Intel Xeon CPUs running at 2.4 GHz. Nonetheless, for the data sets we used in our experiments, up to 8 GBytes of memory were required.

Data sets. We used two real-world data sets in our experiments: Cora[3] and NC Voter [13]. The Cora data set contains 1,879 machine learning publications, which is publicly available, as well as its ground truth. The NC Voter data set is a large voter registration data set from North Carolina, USA. It contains more than 2 million records about voters' information, such as *first name*, *last name*, *gender* and *race*. We have extracted 292,892 records from the original data set, in which 30,000 records with the ground truth labels were used for the experiments on blocking quality, while the whole set of records was used for the experiments on blocking efficiency. The reason why we used 30,000 records for the experiments on blocking quality is to facilitate the comparison with the state-of-the-art blocking techniques discussed in Christen's survey paper [12]. When using the whole set of records of NC Voter, some of these state-of-the-art blocking techniques, e.g., *threshold based string-map blocking* [25], were prohibitively slow to generate the blocks in our experiment environment.

Evaluation measures. We used four common measures to evaluate the quality of blocking: *Pair Completeness* (PC), *Pair quality* (PQ), *Reduction Ratio* (RR) and *F-Measure* (FM) [12], [17], [26]. Let B be the set of blocks generated by applying a blocking method over records in R, $P(S) = \{(t_1, t_2)|t_1, t_2 \in S, t_1 \neq t_2\}$, Γ be a set of distinct pairs in B (i.e., $|\Gamma| = |\bigcup_{b \in B} P(b)|/2$), Γ_{tp} be a set of distinct pairs in Γ that represent the same entity, Γ_m be a collection of all (possibly redundant) pairs in B (i.e., $|\Gamma_m| = \sum_{b \in B} \frac{|b| \times (|b|-1)}{2}$), Ω be a set of all distinct pairs in the entire data set (i.e., $|\Omega| = |P(D)|/2$), and Ω_{tp} be a set of record pairs in Ω that represent the same entity. Then we have: PC=$\frac{|\Gamma_{tp}|}{|\Omega_{tp}|}$, PQ=$\frac{|\Gamma_{tp}|}{|\Gamma|}$, RR=$1 - \frac{|\Gamma|}{|\Omega|}$ and FM=$\frac{2 \times PC \times PQ}{PC + PQ}$. PC measures the degree to which blocks retain true matches, PQ measures the percentage of true matches in the pairs of the blocks, RR measures the degree to which blocks reduce pair comparisons, and FM is the harmonic mean of PC and PQ.

6.1 Blocking Parameters

We calculated the Jaccard similarity distribution of the exact values and q-grams with q=2, 3, 4 of true matches in the two data sets. The textual similarity distributions of the Cora and NC Voter data sets are shown in the upper-left and upper-right subgraphs of Fig. 6, respectively. The textual similarity distribution of the Cora data set was measured

3. http://www.cs.umass.edu/∼mccallum/

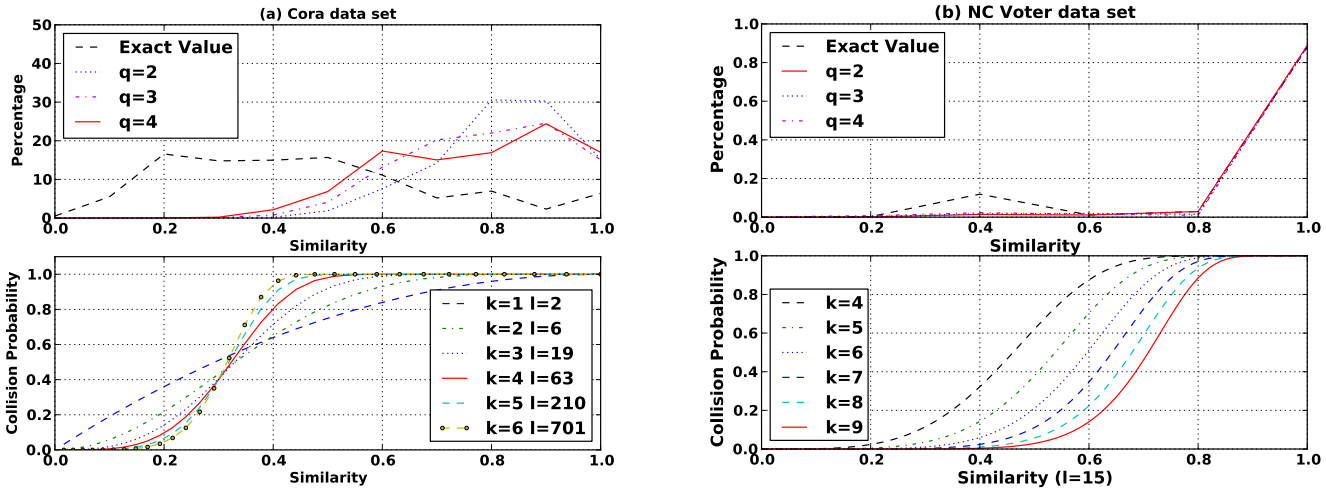


Fig. 6. The textual similarity distribution under different q values, and the collision probability under different k and l values

TABLE 1
An example of patterns based on missing values in three attributes *journal*, *booktitle* and *institution* from Cora

Patterns	Attributes			Concepts
	journal	*booktitle*	*institution*	
1	NOT NULL	NOT NULL	NOT NULL	C_3, C_4, C_6
2	NOT NULL	NOT NULL	NULL	C_3, C_4
3	NOT NULL	NULL	NOT NULL	C_3, C_6
4	NOT NULL	NULL	NULL	C_3
5	NULL	NOT NULL	NOT NULL	C_4, C_7, C_8
6	NULL	NOT NULL	NULL	C_4
7	NULL	NULL	NOT NULL	C_7, C_8
8	NULL	NULL	NULL	C_1

using the values of two attributes *authors* and *title* of the publication records, while the textual similarity distribution of the NC Voter data set was obtained according to the values of two attributes *first name* and *last name* of the voter records. Following the principle of deciding γ-robustness of similarity metrics, we set $q = 4$ for the Cora data set, and $q = 2$ for the NC Voter data set.

The parameters k and l for the Cora data set were determined in terms of the collision probabilities shown in the lower-left subgraph of Fig. 6 and an error ratio $\epsilon = 5\%$. By $\int_0^{s_h} f_s(x)d = \epsilon$ as discussed previously, we have $s_h = 0.3$. Then l and k were tuned such that the collision probability of records whose textual similarity is greater than 0.3 is at least 40% under the error ratio 5%, and on the other hand, to control the percentage of true non-matches in the same block, we chose $s_l = 0.2$ such that any records whose textual similarity is less than 0.2 should only have less than 10% probability of being placed into the same block. The distance between $s_l = 0.2$ and $s_h = 0.3$, together with the required probabilities, determines $k \geq 4$ and $l \geq 63$ for the Cora data set. Considering the time and space efficiency of generating blocks, we chose $k = 4$ and $l = 63$. Our experimental results confirm that these parameters yield the desired results (i.e., the PQ value is best and the PC value is close to the best among $k = 1, \ldots, 6$, as will be illustrated in Fig. 9). Analogously, we chose $k = 9$ and $l = 15$ for the NC Voter data set. Because most of matches whose textual similarity is greater than 0.8, the choice of $k = 9$ and $l = 15$ leads to at least 90% probability of placing two records with 0.8 textual similarity into the same block. The collision probabilities for $k = 4, 5, 6, 7, 8, 9$ and $l = 15$ are depicted in the lower-right subgraph of Fig. 6.

6.2 Semantic Features

In our experiments over the Cora data set, we used the bibliographic taxonomy tree shown in Fig. 3 and a semantic function based on patterns of missing values to define the semantic interpretation of records. As a result, we have 5 bit semantic signature for each record in Cora. Table 1 presents an example of such patterns, which consider missing values in three attributes *journal*, *booktitle* and *institution* of Cora. In practice, missing values may just be empty strings and not necessarily be NULL. But for convenience of expression, we use NULL and NOT NULL refer to missing values and non-missing values, respectively. Take the second pattern in Table 1 for example, it describes that if a record has values in *journal* and *booktitle* but the value of *institution* is missing, then this record is related to the concepts C_3 and C_4 in t_{bib}. The set of patterns described in Table 1 is also *complete* in the sense that every record in Cora can be specified by one of the patterns.

For the NC Voter data set, we built a taxonomy tree upon the meta-data for *race* and *gender*, and defined a semantic function based on the values in the attributes *race* and *gender*, which have uncertain values like 'u'. As a result, we have a 12 bit semantic signature for each record in NC Voter. Due to the existence of uncertain semantic features, performing the semantic-aware LSH blocking is a trade-off decision between PC, and the other two measures (i.e., PQ and RR). The general idea is to increase the PQ and RR values as much as possible, while maintaining the decrease of the PC value within a tolerable range. The details are further discussed in the rest of this section, along with the comparison on blocking quality under different w-way semantic hash functions.

6.3 Blocking Quality

We have conducted experiments to evaluate the quality of blocking from four aspects: (1) comparison of using different

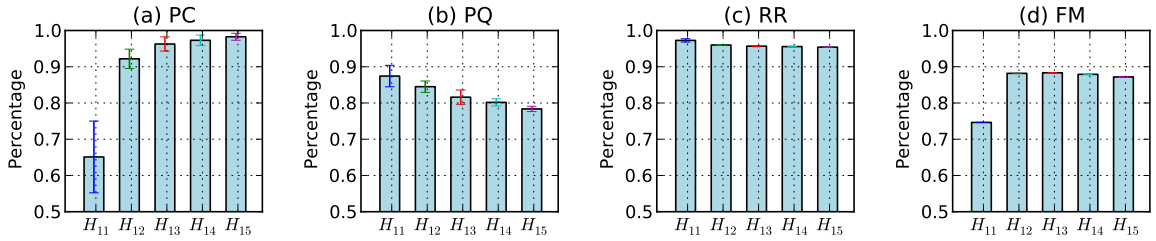


Fig. 7. Experimental results for using different semantic hash functions over the Cora data set (H_{11}: [w=2, $\mu = \wedge$]; H_{12}: [w=1, $\mu = \wedge$ or $\mu = \vee$]; H_{13}: [w=2, $\mu = \vee$]; H_{14}: [w=3, $\mu = \vee$]; H_{15}: [w=4, $\mu = \vee$]), where w indicates the number of randomly chosen hash functions, μ indicates the way of constructing these hash functions, $k = 4$ and $l = 63$

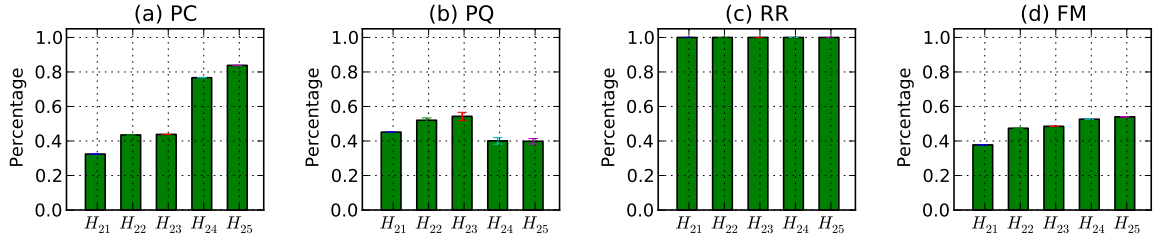


Fig. 8. Experimental results for using different semantic hash functions over the NC Voter data set (H_{21}: [w=1, $\mu = \wedge$ or $\mu = \vee$]; H_{22}: [w=3, $\mu = \vee$]; H_{23}: [w=5, $\mu = \vee$]; H_{24}: [w=7, $\mu = \vee$]; H_{25}: [w=9, $\mu = \vee$]), where w indicates the number of randomly chosen hash functions, μ indicates the way of constructing these hash functions, $k = 9$ and $l = 15$

semantic hash functions; (2) comparison of using basic LSH and semantic-aware LSH; (3) comparison of using different taxonomy trees; (4) comparison with existing blocking techniques.

6.3.1 Comparison of Semantic Hash Functions

We evaluated the effects of using different semantic hash functions on the quality of blocking. Fig. 7 and Fig. 8 present the results of incorporating five different semantic hash functions into the LSH blocking for textual similarity over the Cora and NC Voter data sets, respectively.

For both data sets, the PC values decrease when w increases in the case $\mu = \wedge$, while the PC values increase when w increases in the case $\mu = \vee$. This is consistent with the collision probability of semantic hash functions shown in Fig. 5. For the PQ values, the two data sets have different results in terms of the semantic hash functions used in our experiments, which were due to the different characteristics of the data sets, including their textual similarity distributions shown in Fig. 6 and semantic features. For the Cora data set, the PQ values always increase when w increases and $\mu = \wedge$, and decrease when w increases and $\mu = \vee$. This indicates that records with higher degrees of semantic similarities mostly refer to true matches. For the NC Voter data set, however, due to the significant amount of uncertain values in *race* and *gender*, using semantic hash functions to find records with higher semantic similarity may also lead to the addition of non-matches into the same blocks, thus decreasing the PQ values. The RR values have the same trends in both data sets, i.e., slightly decreasing when the collision probabilities of finding semantically similar records increase. The RR values for the NC Voter data set also indicate that RR is not an informative measure when a data set is large and relatively clean. From the FM values in Fig. 7 and Fig. 8, we can conclude that, for both data sets, the overall performance of PC and PQ goes stable when $\mu = \vee$ and w is greater than 50% of the total number of semantic signatures.

6.3.2 Comparison of LSH and SA-LSH

We conducted an experiment to compare the blocking quality of using the LSH blocking that only considers textual similarity (written as LSH) and using the semantic-aware LSH blocking (written as SA-LSH). Fig. 9 show the blocking results of using LSH and SA-LSH methods over the Cora and NC Voter data sets, respectively. For the SA-LSH methods in Fig. 9, we used the lowest threshold for semantic similarity, i.e., two records are semantically similar if their semantic similarity is greater than $1/5$ in the Cora data set (resp. $1/12$ in the NC Voter data set).

In Fig. 9 (a), the PC values of LSH increase to 97% and above when k increases to 3, and the PC values of SA-LSH increase correspondingly but are lower than the PC values of LSH. The gap between the PC values of LSH and SA-LSH is correlated with the degree of noisiness in semantic features. For example, by Fig. 9 (a), we know that the semantic features of the Cora data set are noisy. This is because some records in Cora do not comply with any patterns in Table 1. In Fig. 9 (d), the PC values of LSH and SA-LSH are the same. This is due to the fact that the semantic features of the NC voter data set is not noisy, although they may contain uncertain values. Fig. 9 (b) and Fig. 9 (e) show that SA-LSH methods can considerably improve the PQ values of LSH in both data sets, where Fig. 9 (b) uses a 0-1 scale and Fig. 9 (e) uses a 0-0.4 scale. For the Cora data set, both LSH and SA-LSH methods have the highest PQ value at $k = 4$. This is because the PC values reach almost 1 at $k = 4$, and increasing k leads to reducing true matches within the same blocks. For the NC Voter data set, the SA-LSH methods always have the higher PQ values when the k value increases. This is because that their corresponding PC values are far below 1. We choose $k = 4$

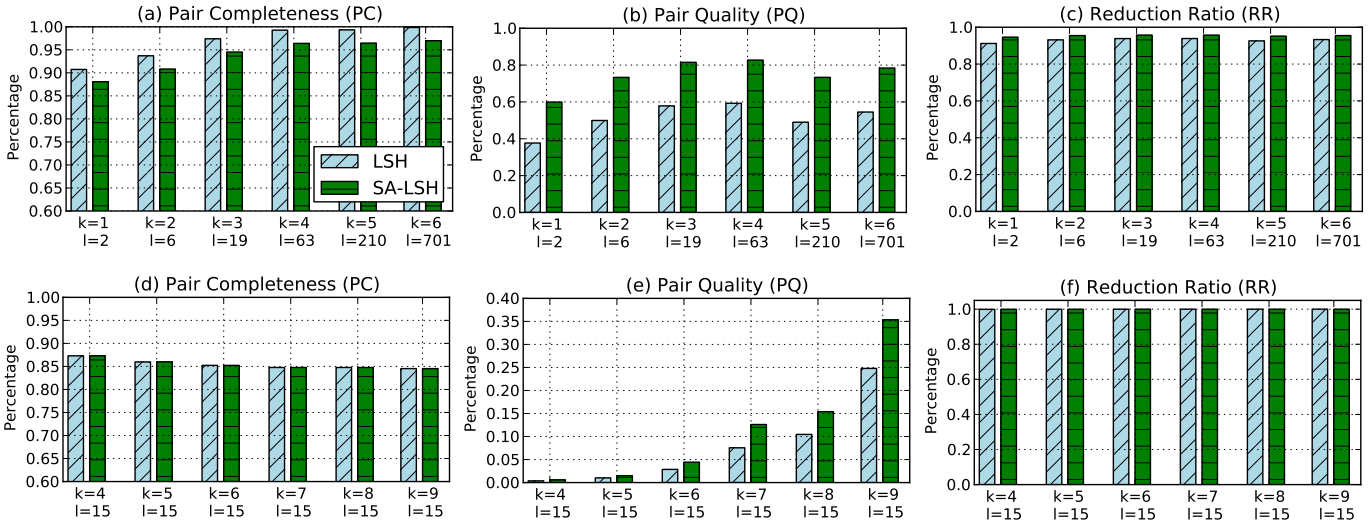


Fig. 9. Experimental results for comparing LSH and SA-LSH: (a)-(c) over the Cora data set, and (d)-(f) over the NC Voter data set, where LSH refers to basic LSH only over textual similarity space, and SA-LSH refers to semantic-aware LSH over textual and semantic similarity spaces

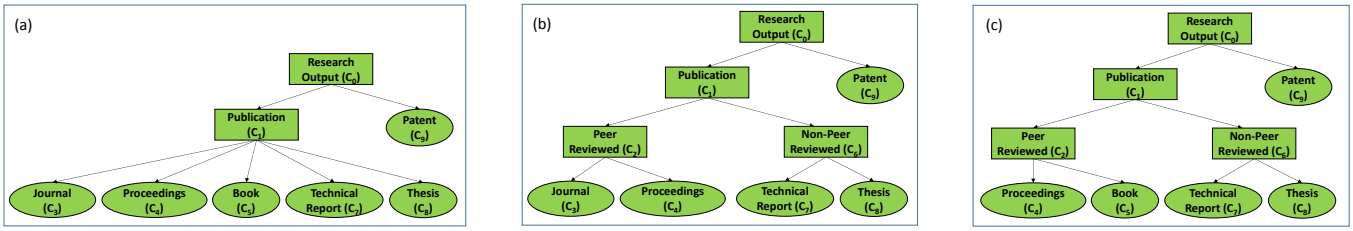


Fig. 10. Three variants of the bibliographic taxonomy tree t_{bib}: (a) $t_{(bib,1)}$ (left), (b) $t_{(bib,2)}$ (center), and (c) $t_{(bib,3)}$ (right)

and $l = 63$ in Fig. 9 (a)-(c), and $k = 9$ and $l = 15$ in Fig. 9 (d)-(f), which experimentally verified the validity of our choices on blocking parameters as discussed in Section 6.2. For both data sets, SA-LSH methods have higher RR values than the corresponding LSH methods. This is because SA-LSH methods can eliminate pairs that are textually similar but semantically dissimilar, whereas LSH methods fail to filter out these pairs. Nevertheless, because the NC Voter data set is large and relatively clean, the gap of the RR values between LSH and SA-LSH is not obvious.

6.3.3 Comparison of Taxonomy Trees

To investigate the impacts of taxonomy trees on blocking results, we conducted an experiment that takes into account the variants of taxonomy trees, and compares their blocking results. In particular, our focus was on examining how the structural changes on a taxonomy tree (e.g., missing concepts) may affect the quality of blocks. The experiment was performed over the Cora data set with the same parameter setting as described in Section 6.3.2, and we have used three variants of taxonomy trees as described in Fig. 10.

Table 2 describes the impacts of applying SA-LSH on the blocking results generated by LSH when using different taxonomy trees $t_{(bib,i)}$ $(i = 1, 2, 3)$. We can see that the PC values always decrease and the PQ, RR and FM values always increase after incorporating semantic features into the LSH blocking process. Nevertheless, for these three variants (i.e., $t_{(bib,1)}$ removes *Peer Reviewed* and *Non-Peer Reviewed* from t_{bib}, $t_{(bib,2)}$ misses *Book* from t_{bib}, and $t_{(bib,3)}$ misses *Journal* from t_{bib}), the decrease of the PC values is less than

TABLE 2
Experimental results for comparing the impact on blocks over Cora using different taxonomy trees

	Taxonomy trees			
	t_{bib}	$t_{(bib,1)}$	$t_{(bib,2)}$	$t_{(bib,3)}$
PC	-3.55±0.59	-3.32±0.30	-3.05±0.18	-3.02±0.21
PQ	+24.75±3.91	+23.52±2.40	+23.23±2.80	+14.01±1.48
RR	+2.24±0.54	+2.20±0.24	+2.07±0.31	+1.41±0.19
FM	+16.26±3.42	+15.82±1.72	+15.40±2.17	+9.29±1.16

the decrease of the PC value using t_{bib}. This is because the records that are originally related to missing concepts have been changed to relate with their parent concepts, which increases the semantic relatedness among records and helps capture true matches that were not semantically related in t_{bib}. In addition to this, the three variants of t_{bib} also have different impacts on PQ. For $t_{(bib,1)}$, it has a better PQ value than $t_{(bib,2)}$ and $t_{(bib,3)}$. This is because some records in Cora cannot be clearly identified as peer-reviewed or non-peer-reviewed based on patterns of missing information, so removing *Peer Reviewed* and *Non-Peer Reviewed* only slightly decreases PQ. For $t_{(bib,2)}$ and $t_{(bib,3)}$, because the number of records originally related to *Book* is much smaller than the number of records originally related to *Journal* in t_{bib}, the PQ value of $t_{(bib,2)}$ is thus better than the PQ value of $t_{(bib,3)}$. Compared with the results of only applying LSH, the RR values of applying SA-LSH with $t_{(bib,i)}$ $(i = 1, 2, 3)$ generally increase. However, the degree of increases varies in terms of missing concepts. In the case of more missing concepts, more records become semantically related through the parent concepts of missing concepts, regardless whether

TABLE 3
The state-of-the-art blocking techniques used in our comparison

Blocking technique	Abbrev.	No. of parameter settings		Time (sec)	No. of candidate pairs
		Cora	NC Voter	NC Voter (the best-performing setting for FM)	
Traditional blocking [18]	TBlo	1	1	0.9088	15,272
Array based sorted neighbourhood [21], [22]	SorA	5	5	1.0097	29,999
Inverted index based sorted neighbourhood [10]	SorII	5	5	1.1828	58,549
Adaptive sorted neighbourhood [41]	ASor	8	8	2.4507	15,272
Q-gram based indexing [6]	QGr	4	4	4.9046	15,288
Threshold based canopy clustering [32]	CaTh	8	8	13.7577	15,333
Nearest neighbour based canopy clustering [10]	CaNN	8	8	70.3243	151,804
Threshold based string-map blocking [25]	StMT	32	30	2197.4365	132,896
Nearest neighbour based string-map blocking [1]	StMNN	32	32	1662.5243	84,002
Suffix-array based blocking [2]	SuA	6	6	1.7224	23,305
Suffix-array based blocking using all sub-strings [2]	SuAS	6	6	2.2947	36,876
Robust suffix-array blocking [15]	RSuA	48	48	3.6113	23,305
Locality-sensitive-hashing based blocking	LSH	1	1	2.340	5,110
Semantic-aware locality-sensitive-hashing	SA-LSH	1	1	3.872	3,565

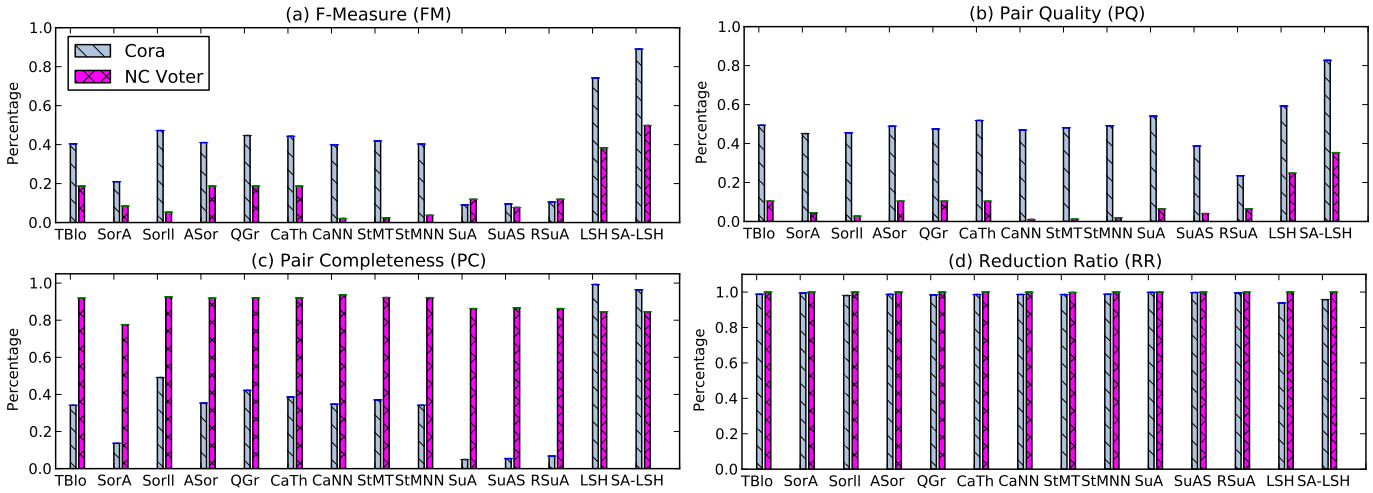


Fig. 11. Experimental results for comparing LSH and SA-LSH (k=9, l=15) with the state-of-the-art blocking techniques.

or not they represent the same entities.

6.3.4 Comparison to State-of-the-Art

We conducted an experiment to compare the quality of our proposed blocking methods with the state-of-the-art blocking techniques discussed in Christen's survey paper [12]. Table 3 depicts these state-of-the-art blocking techniques, their abbreviations, the number of parameter settings that were evaluated over the data sets, and the average time taken for building blocks over NC Voter.

Following the experimental setup in [12], we used a total of 163 parameter settings for Cora and 161 parameter settings for NC Voter. The reason why NC Voter has a less number of parameter settings than Cora is that 2 of the 163 parameter settings (i.e., StMT with the string similarity *bigram*, thresholds $\{0.95/0.85, 0.9/0.8\}$, the grid size 1000 and mapping dimension 15) failed to generate any blocking results over the NC Voter data set, but worked well over the Cora data set. More specifically, a blocking key on *authors* and *title* was defined for the Cora data set, and similarly, a blocking key on *first name* and *last name* was defined for the NC Voter data set. For SorA and SorII, the window size was set to $\{2, 3, 5, 7, 10\}$. For ASor, RSuA, StMT and StMNN, the string similarity functions *Jaro-Winkler*, *bigram*, *edit-distance* and *longest common substring* [12] were used. For ASor, RSuA and QGr, similarity thresholds were set to $\{0.8.0.9\}$. For SuA, SuAS and RSuA, the minimum suffix length and maximum block size were set to $\{3, 5\}$ and $\{5, 10, 20\}$, respectively. For all blocking techniques using q-grams, q was set to $\{2, 3\}$. For CaTh and CaNN, the *Jaccard* and *TF-IDF cosine* similarities were used, in combination with the thresholds $\{0.9/0.8, 0.8/0.7\}$ for CaTh and $\{5/10, 10/20\}$ for CaNN. For StMT and StMNN, the grid size was set to $\{100, 1000\}$, and the mapping dimension to $\{15, 20\}$. The thresholds were set to $\{0.95/0.85, 0.9/0.8\}$ for CaTh and $\{5/10, 10/20\}$ for CaNN. All blocking techniques presented in Table 3 were implemented in Python.

Fig. 11 describes the experimental results of comparing LSH and SA-LSH with the blocking techniques listed in Table 3. For each blocking technique, the result with the best-performing parameter setting is presented. We can see that the FM values of LSH and SA-LSH are much better than the others. The PQ values of LSH and SA-LSH are higher than the others over both data sets, while a number of blocking techniques such as CaNN, StMT and StMNN have very low PQ values (less than 1%) for the NC Voter data set. For the PC values, although LSH and SA-LSH performed much better than the others over the Cora data set, their PC values were slightly lower over the NC Voter data set. The RR values of all blocking techniques are quite close.

We have conducted an experiment to verify how effectively using semantic features can improve the LSH

blocking in comparison with the meta-blocking method introduced in [37]. Four pruning algorithms (i.e., WEP, CEP, WNP, and CNP) and five weighting schemes (i.e., ARCS, CBS, ECBS, JS, and EJS) have been proposed in [37], and the authors have implemented them in Java. Fig. 12 presents the experimental results, where the result of each pruning algorithm is taken from a weighting scheme with the highest FM* value, and the parameter settings for SA-LSH are the same as in Fig. 11. Note that, PQ*=$\frac{|\Gamma_{tp}|}{|\Gamma_m|}$ is used in [37], which is different from PQ used by us and some others [12], [26], and accordingly, FM*=$\frac{2\times PC\times PQ*}{PC+PQ*}$. From Fig. 12, we can see that WNP+JS and WEP+ARCS have the highest FM* values for the Cora and NC Voter data sets, respectively. Thus, the meta-blocking method performs better than SA-LSH in terms of FM* values. Nevertheless, our experiments show that SA-LSH has the highest PC value over Cora, and has the same highest PC value with several combinations of prune algorithms and weighting schemes over NC Voter.

6.4 Blocking Efficiency and Scalability

We have also performed experiments to explore: (a) the efficiency of the LSH and SA-LSH methods in comparison with other blocking techniques; (b) the scalability of the LSH and SA-LSH methods in terms of the increasing numbers of records in data sets. For the task (a), we used the same NC Voter data subset as used in Section 6.3, which has 3,000 records. For the task (b), in addition to using the NC Voter data subset in Section 6.3 and the full NC Voter data set with 292,892 records as test data sets, we also created another six test data sets of different sizes which contain 10,000, 50,000, 100,000, 150,000, 200,000 and 240,000 records from the full NC Voter data set, respectively. Then we ran our experiments five times for each test data set, and took their average runtime used in the blocking process.

In Table 3, the blocking time and number of candidate pairs for each blocking technique were taken from the best-performing result in the sense that its FM value is the highest one among the results in all parameter settings. We can see that the times vary significantly amongst the state-of-the-art blocking techniques (i.e., from 0.9088 seconds for TBlo to 2197.4365 seconds for StMT). The best-performing results for both StMT and StMNN have the longest time to build blocks. In comparison with these, the blocking times of LSH and SA-LSH are 2.340 and 3.872 seconds, respectively, which include the time for constructing the taxonomy tree and building the semantic function (i.e., mappings from the records in the NC Voter data set to the related concepts in the taxonomy tree).

Fig. 13 presents the PC, PQ, RR and scalability results of LSH and SA-LSH, in which the horizontal axis indicates the sizes of data sets, and SF in Fig. 13 (d) refers to the process of building the semantic function, including the construction of the taxonomy tree. From Fig. 13 (a)-(c), we can see that LSH and SA-LSH have almost same PC values, which again indicates that the semantic features of these NC Voter data sets are not noisy. Nevertheless, the PQ values vary in different data sets, and the PQ values of SA-LSH are always significantly higher than the PQ values of LSH. For the RR values, both LSH and SA-LSH have the value 0.9999 over all data sets. Fig. 13 (d) presents the times required by LSH and SA-LSH over data sets of different sizes, and their corresponding times of constructing the taxonomy tree and building the semantic function. Three dashed lines are the trendlines added for indicating their scalability.

7 Conclusion

In this paper we have developed the semantic-aware LSH blocking framework that takes into account both textual and semantic similarities in the ER blocking process. Our experimental results show that semantic information can be leveraged to improve the blocking quality, and the integration of textual similarity and semantic similarity with the LSH technique provides us an efficient and scalable blocking technique for ER with improved quality.

In the future, we plan to extend our methods to handling heterogenous data sets by leveraging context information and knowledge reasoning tools within a network environment. In particular, we will investigate the mining and learning methods for discovering semantic features. This task is important for increasing the applicability and efficiency of our proposed semantic-aware LSH blocking framework to real-world ER problems. We envision developing knowledge bases for given ER tasks, which contain the semantic features discovered through the mining and learning process.

Acknowledgment

The authors would like to thank Peter Christen at Australian National University for his valuable comments, and sharing the evaluation code used in his blocking survey [12].

References

[1] N. Adly. Efficient record linkage using a double embedding scheme. In *DMIN*, pages 274–281, 2009.
[2] A. Aizawa and K. Oyama. A fast linkage detection scheme for multi-source information integration. In *WIRI*, pages 30–39, 2005.
[3] A. Andoni and P. Indyk. Near-optimal hashing algorithms for near neighbor problem in high dimension. *Communications of the ACM*, 51(1), 2008.
[4] A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *SIGMOD*, pages 783–794, 2010.
[5] M. Bawa, T. Condie, and P. Ganesan. LSH forest: self-tuning indexes for similarity search. In *WWW*, pages 651–660, 2005.
[6] R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage. In *ACM SIGKDD*, volume 3, pages 25–27, 2003.
[7] M. Bilenko, B. Kamath, and R. J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *ICDM*, pages 87–96, 2006.
[8] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *JCSS*, 60(3):630–659, 2000.
[9] F. Chierichetti and R. Kumar. LSH-preserving functions and their applications. In *SIAM 2012*.
[10] P. Christen. Towards parameter-free blocking for scalable record linkage. *Technical Report TR-CS-07-03*.
[11] P. Christen. *Data matching*. Springer, 2012.
[12] P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *TKDE*, 24(9):1537–1555, 2012.
[13] P. Christen. Preparation of a real voter data set for record linkage and duplicate detection research. Technical report, Australian National University, 2013.
[14] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *ACM SIGKDD*, pages 475–480, 2002.
[15] T. De Vries, H. Ke, S. Chawla, and P. Christen. Robust record linkage blocking using suffix arrays. In *CIKM*.
[16] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to match ontologies on the semantic web. *The VLDB Journal*, 12(4):303–319, 2003.

(a) Cora data set

Method	Initial blocks		Final blocks			
	PC	PQ*	Weight	PC	PQ*	FM*
WEP	0.999	0.0497	JS	0.867	0.749	0.803
CEP			EJS	0.188	0.933	0.313
WNP			JS	0.921	0.368	0.526
CNP			CBS	0.193	0.515	0.281
SA-LSH			–	0.969	0.148	0.257

(b) NC Voter data set

Method	Initial blocks		Final blocks			
	PC	PQ*	Weight	PC	PQ*	FM*
WEP	0.847	0.150	ARCS	0.800	0.504	0.618
CEP			ARCS	0.847	0.152	0.258
WNP			ARCS	0.842	0.111	0.196
CNP			JS	0.802	0.244	0.374
SA-LSH			–	0.847	0.360	0.505

Fig. 12. Experimental results for comparing SA-LSH and the meta-blocking method over: (a) Cora (left) and (b) NC Voter (right).

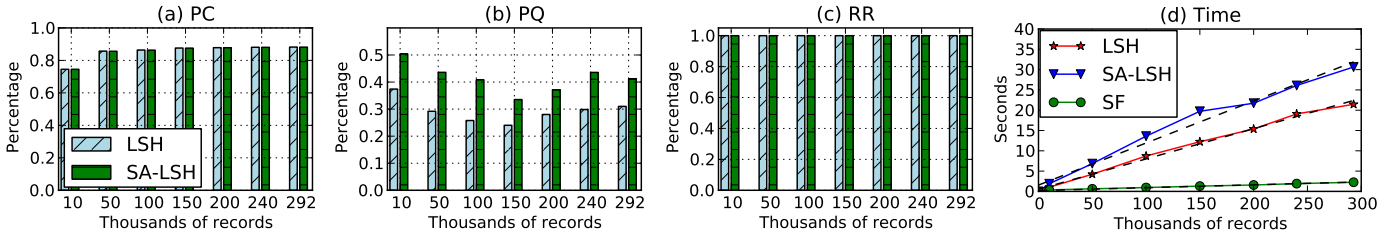


Fig. 13. The PC, PQ, RR and time performance of LSH and SA-LSH (k=9, l=15) over the NC Voter data sets of different sizes.

[17] M. G. Elfeky, V. S. Verykios, and A. K. Elmagarmid. Tailor: A record linkage toolbox. In *ICDE*, pages 17–28, 2002.

[18] I. Fellegi and A. Sunter. A theory for record linkage. *J AM STAT ASSOC*, 64(328):1183–1210, 1969.

[19] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *IJCAI*, pages 1606–1611, 2007.

[20] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB, 1999*.

[21] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *ACM SIGMOD Record*, volume 24.

[22] M. A. Hernández and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data mining and knowledge discovery*, 2(1):9–37, 1998.

[23] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.

[24] E. Ioannou, O. Papapetrou, D. Skoutas, and W. Nejdl. Efficient semantic-aware detection of near duplicate resources. In *ESWC*, pages 136–150. 2010.

[25] L. Jin, C. Li, and S. Mehrotra. Efficient record linkage in large data sets. In *DASFAA*, pages 137–146, 2003.

[26] M. Kejriwal and D. P. Miranker. An unsupervised algorithm for learning blocking schemes. In *ICDM*, pages 340–349, 2013.

[27] B. Kenig and A. Gal. MFIBlocks: An effective blocking algorithm for entity resolution. *Information Systems*, 38(6):908–926, 2013.

[28] H.-S. Kim and D. Lee. HARRA: fast iterative hashed record linkage for large-scale data collections. In *EDBT*, pages 525–536, 2010.

[29] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In *VLDB, 2007*.

[30] A. Maedche and S. Staab. Measuring similarity between ontologies. In *EKAW*, pages 251–263. 2002.

[31] P. Malhotra, P. Agarwal, and G. Shroff. Graph-parallel entity resolution using LSH & IMM. In *EDBT/ICDT Workshops*, pages 41–49, 2014.

[32] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *SIGKDD, 2000*.

[33] G. Miller and C. Fellbaum. Wordnet: An electronic lexical database, 1998.

[34] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In *SODA*, pages 1186–1195, 2006.

[35] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl. Eliminating the redundancy in blocking-based entity resolution methods. In *JCDL*, pages 85–94, 2011.

[36] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl. Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data. In *WSDM*, pages 53–62, 2012.

[37] G. Papadakis, G. Koutrika, T. Palpanas, and W. Nejdl. Meta-blocking: Taking entity resolution to the next level. *IEEE TKDE*, 26(8):1946–1960, 2014.

[38] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI, 1995*.

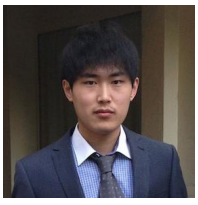
[39] S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. Entity resolution with iterative blocking. In *SIGMOD 2009*.

[40] W. Wu, H. Li, H. Wang, and K. Zhu. Towards a probabilistic taxonomy of many concepts. Technical report, Technical Report MSR-TR-2011-25, Microsoft Research, 2011.

[41] S. Yan, D. Lee, M.-Y. Kan, and L. C. Giles. Adaptive sorted neighborhood methods for efficient record linkage. In *JCDL*.

[42] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity search: the metric space approach*, volume 32. Springer, 2006.

Qing Wang received her Ph.D. in Computer Science from Christian-Albrechts-University Kiel, Germany, in 2010. She has worked in the IT industry for over ten years. From 2009 to 2012, she worked as an Information System Analyst at the University of Otago, New Zealand. In April 2012, she joined the Research School of Computer Science, Australian National University. Her research interests are *data management*, *data mining*, *conceptual modelling* and *knowledge reasoning*.



Mingyuan Cui was a postgraduate student at the Research School of Computer Science, The Australian National University, Australia. He received his Bachelor of Engineering from Wuhan University, China, in 2013, and his Masters degree in Computing (Honours) from The Australian National University, Australia, in 2015. His research interest is in the areas of *data mining* and *machine learning*.

Huizhi Liang received her Ph.D in Computer Science from Queensland University of Technology, Australia, in 2011. She worked as a postdoctoral researcher at the Research School of Computer Science, Australian National University, during 2012-2014, and joined the Department of Computing and Information Systems, University of Melbourne, in September 2014. Her research interests include *recommender systems*, *data mining* and machine learning.