Towards a Scalable and Robust Entity Resolution -Approximate Blocking with Semantic Constraints

Mingyuan Cui

Supervisor: Dr. Qing Wang, Dr. Huizhi Liang

COMP8740: Artificial Intelligence Project

Australian National University Semester 1, 2014

May 30, 2014

Acknowledgements

First and foremost, I would like to thank my supervisors, Dr. Qing Wang and Dr. Huizhi Liang. Without their assistance and dedicated involvement in every step throughout the process, this report would have never been accomplished. I would like to thank them very much for their support and understanding over the past months.

I would like to thank Dr. Weifa Liang for his helpful guidance on the whole project and technical writing. I also would like to thank Dr. Peter Christen for his kind guidance to my presentation. His book helps a lot.

Most importantly, none of this could have happened without the understanding and support of my family and my girlfriend. This report stands as a testament to their unconditional love and encouragement.

Abstract

Entity resolution, or record linkage, is the process that identifies data records over one or more datasets which refer to the same real world entity. To deal with large datasets, many real-life applications require scalable and high-quality entity resolution techniques. Blocking techniques can help to scale-up the entity resolution process. Locality sensitive hashing (LSH) is an approximate blocking approach that hashes objects within a certain distance into the same block with high probability. This technique can filter out records with low similarities, thus decreases the number of comparisons. However, the traditional approach only considers the textual or string similarity of records while the semantic similarity or constraints of records are ignored. This project is to propose and implement a framework that incorporates semantic constraints into the approximate blocking process to achieve scalable, high performance entity resolution. Firstly, minhashing based locality sensitive hashing methods are applied to generate minhash signatures based on the textual similarity of records. Then, for the semantic constraints, the whole domain knowledge of a dataset is extracted into a domain tree. After applying constraints functions according to a set of pre-set rules, a set of semantic signatures are generated. Then these two sets of signatures are combined to group the records into blocks. The experiments are conducted based on the Cora dataset. The results show that this framework makes blocking much more accurate, and in the meanwhile keeps high completeness.

Keywords: Entity Resolution, Blocking, Locality Sensitive Hashing, Constraints, Semantic Tree

Contents

Acknowledgements **1**

Abstract **2**

1 Introduction **5**
1.1 Objectives 6
1.2 Motivating Example 6
1.3 Outline 7

2 Background and Related Work **8**
2.1 Entity Resolution 8
2.2 ER Blocking 9
2.2.1 Sorted Neighborhood Blocking 9
2.2.2 Q-gram Based Blocking 9
2.2.3 Canopy Clustering 10
2.3 Locality Sensitive Hashing 10
2.4 Semantic Constraints 10

3 Approximate Blocking with Semantic Constraints **11**
3.1 Overview 11
3.2 Locality Sensitive Hashing 12
3.2.1 Jaccard Similarity and Jaccard Distance 12
3.2.2 Minhashing 13
3.2.3 Locality Sensitive Hashing 15
3.3 Semantic Constraints 18
3.3.1 Domain Tree 18
3.3.2 Semantic Interpretations 19
3.3.3 Semantic Signatures 20
3.4 LSH Blocking with Semantic Constraints 21

3.4.1 Logical Conjunction 21
3.4.2 Logical Disjunction 22

4 Evaluation **24**
4.1 Experiment Design 24
4.1.1 Dataset 24
4.1.2 Metrics 25
4.1.3 Compared Methods 27
4.1.4 Experimental Environment 27
4.2 Experimental Results 27
4.2.1 Parameter Setting 27
4.2.2 Comparison of LBS to LB 29
4.2.3 Comparison of LBS with Different Constraints 33
4.2.4 Block Size Distribution 35
4.3 Summary 37

5 Conclusion **38**

Appendices **39**

Chapter 1

Introduction

Entity resolution [6] (ER) is the task of identifying records over one or more datasets that refer to the same entities in the real world. Traditional entity resolution techniques compute the similarities between all pairs of records, which would be very expensive for large datasets. For example, if an entity resolution is performed on a dataset that consists of 30000 records, then the number of pairwise comparisons is large than 4 hundred million, which would take much time.

Some techniques have been proposed to make entity resolution scalable. *Dataset blocking* [6] is a technique that enables real-time entity resolution by blocking a dataset into smaller blocks. In this way entity resolution only needs to compare records within the same block, avoiding a considerable number of unnecessary pairwise comparisons. This method is actually a trade-off between effectiveness and efficiency within an acceptable range, and the weights between them are adjusted by specific related parameters in the blocking process. This kind of similarity-aware blocking improves the performance of traditional indexing by pre-calculating similarities of attribute values.

Blocking methods usually are simply based on textual similarities, setting some thresholds and filtering out the pairs that are unlike to be linked, which is on the basic textual level. Usually such methods can not give accurate results, or they may sacrifice the completeness for more accurate results. This is because in reality, different entities sometimes are distinguished not only by their external characteristics but also by their internal natures. Therefore I consider to incorporate *semantic constraints* according to related domain knowledge into blocking process, which will lead to considerable effects on the effectiveness of blocking process.

My project focuses on the effects of the constraints on ER blocking. In this report, a two-level blocking method which combines textual similarity with semantic con-

straints is proposed. On the first level, a technique called *locality sensitive hashing* [10] (LSH), which will be introduced in the next chapter, is adopted to approximately block records based on textual similarity and the potential pair matches are preserved. On the second level, semantic constraints are applied to obtain a more accurate result. The proposed two-level approach is evaluated using Cora dataset with and without semantic constraints. Experiments show that the two-level approach significantly improves the accuracy without obvious loss of completeness.

1.1 Objectives

The goal of this project is to develop a framework that incorporates semantic constraints into the blocking process for achieving scalable and robust entity resolution results. The specific objectives are:

1. Investigate existing blocking techniques for entity resolution.
2. Evaluate and incorporate semantic constraints into the blocking precess.
3. Analyze the efficiency and effectiveness of the developed framework over real datasets.

The main features of my framework are:

- Using a domain tree to store the domain knowledge and apply semantic interpretations to interpret the semantics of records from one or more datasets.
- Compressing the textual and semantic features into numeral signatures and using locality sensitive hashing to group record into different blocks.

1.2 Motivating Example

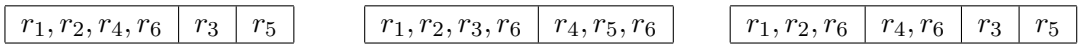
The general purpose of this part is to illustrate the proposed framework using examples. Consider the records in Figure 1.

- based on the textual similarity of the titles and authors, the blocking results is shown in Figure 1.2.a.
- based on the semantic similarity, we know $r1$, $r2$ and $r3$ are conference papers, $r4$ and $r5$ are technical reports, and $r6$ is ambiguous for semantics. Thus the blocking result is as shown in Figure 1.2.b.
- based on the combination of two kinds of similarity, the results is as shown in Figure 1.2.c.

In the first two cases, the results are not satisfying while after consider two kinds of similarity together, the result gets much better.

REC	TITLE	AUTHORS	PUBLISHER
r_1	Cascade-correlation learning architecture	E. Fahlman and C. Lebiere	NISPS
r_2	Cascade correlation learning architecture	E. Fahlman & C. Lebiere	NIS
r_3	Genetic cascade-correlation learning		Neural Ntw.
r_4	The cascade-correlation learning architecture	Fahlman, S., & Lebiere, C.	TR
r_5	Controlled growth of cascade correlation nets		TR
r_6	The cascade-correlation learning architecture	Lebiere, C. and Fahlman, S.	

Figure 1.1: Sample publication records



(a) (b) (c)

Figure 1.2: Sample blocks (a) based on textual similarity; (a) based on semantic similarity; (b) based on both textual similarity and semantic similarity.

1.3 Outline

This project report is structured as follows:

- Chapter 2 presents a general picture for the ER process and the related works for blocking, LSH and semantic constraints.
- Chapter 3 presents the proposed blocking framework in details.
- Chapter 4 presents my experimental results, including a discussion on tuning parameters and selecting rules in the experiments.
- Chapter 5 is the conclusions and future work of the project.

Chapter 2

Background and Related Work

This chapter introduces the concept of entity resolution and related work. At first, a high-level overview of entity resolution is presented by a visual representation. It discusses the main stages of the ER process, and indicates the area, i.e. ER blocking, which my project applies to. Existing ER blocking techniques and related work are also introduced subsequently.

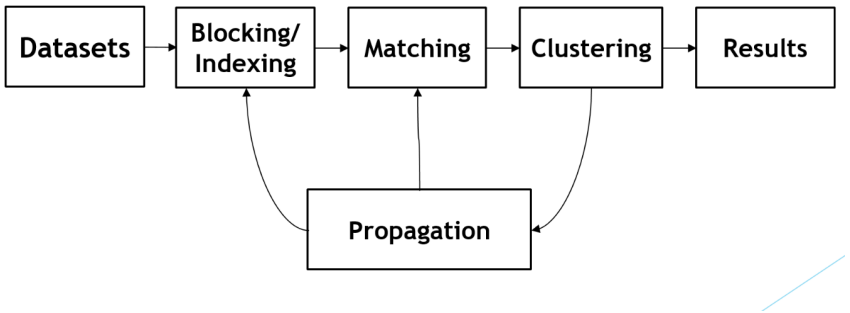


Figure 2.1: High-level overview of entity resolution

2.1 Entity Resolution

Entity resolution is the task of identifying records over one or more datasets that refer to the same entities in the real world. Figure 2.1 presents the main stages of entity resolution.

- Records from one or more datasets first get blocked by ER blocking, or *indexing* [7]. In this stage, records that are likely to refer to the same entity are considered as *candidate records*. In this way, ER blocking minimizes the number of comparisons needed for the following stages.

- *Matching process* [17] makes record comparisons to gives some measurements of how two records are similar.
- *Clustering* [12], or *classification* process decides which records are regarded as the references to the same entity in real world. Various clustering techniques have been developed in the past, such as threshold-based clustering and cost-base clustering [6].
- *Propagation* [20] process reflects the ER results to the previous stages in a new iteration:
    1. Iterating the blocking, matching and clustering stages until a desired ER result is obtained, for example, *HARRA* method [14] and *iterative blocking* [20].
    2. Iterating the matching and clustering stages until a desired ER result is obtained, which is often called *collective ER* [3]

My project focuses on blocking process.

2.2 ER Blocking

This part is aim to introduce some existing techniques for ER blocking. The general approach of blocking techniques is to process all records of the datasets, to insert each of the record into one or more blocks, within which all pairs are regarded as candidate records, which are likely to refer to the same entity. Below are widely used blocking techniques.

2.2.1 Sorted Neighborhood Blocking

Sorted neighborhood blocking [21] used sorting keys to group records. For example, use the value of an attribute as a key, and sort the records according to this sort key, so that candidate records are put close. This method is efficient with a time complexity $O(nlogn)$ but can not handle records that are highly similar but have different keys, such as "Peter Smith" and "Smith Peter".

2.2.2 Q-gram Based Blocking

If datasets are dirty due to spelling errors, duplications or missing values, then it will be difficult to find candidate records. *Q-gram based blocking* [6] can solve this problem. A *q-gram* [7] (also known as n-gram) of a record is a set of contiguous sequence of length q characters from itself. When q equals to 1, this is called a "unigram", and when q equals to 2, then this is called a "bigram" (or, a "digram").

Example 2.1: Consider a record, "university", and we choose q = 3. Then the set of 3-gram for this record is {uni, niv, ive, ver, ers, rsi, sit, ity}.

By splitting strings into small parts, records that shares a part of the value of some attribute can be blocked into the same block. But this method needs to compare all the pair similarities which is computationally expensive.

2.2.3 Canopy Clustering

The blocking is a way of clustering records. Clustering algorithms usually is expensive in time and space complexity, which a blocking process should be efficient. *Canopy clustering* [6] achieves this by calculate *distances* between blocking keys and block records into one or more overlapping clusters. There are many distance measures for clustering, such as *Euclidean distances*, *Jaccard distance*, *cosine distance*, *hamming distance* and *edit distance* [19].

2.3 Locality Sensitive Hashing

LSH was firstly introduced for solving *approximate nearest neighbor search* problem [10]. In the pase years, a couple of LSH families have been proposed for specific distance measures, for example, Jaccard distance [4] and l_s distance [8]. Some variants of LSH have been proposed to improve the quality of the original LSH technique. The entropy-based LSH [18] and multi-probe LSH [15] methods both try to reduce the number of hash functions required by the original LSH method while achieving the same accuracy. The LSH forest method [2] 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 measures. In a word, LSH is an effective technique to be applied into ER process.

2.4 Semantic Constraints

Semantic technologies have great influences in many areas, such as improving search on the web and semantic indexing in information retrieval [9]. Domain knowledge can be modelled in the ontology, which have been extensively studied in the past, and used in a wide range of applications. In a word, constraints allow us to take advantages of domain knowledge for improved ER quality. In previous studies, a variety of constraints have also been explicitly investigated in relating to ER process. Nevertheless, little work has been done on using semantic constraints from domain knowledge for blocking in entity resolution.

Chapter 3

Approximate Blocking with Semantic Constraints

This chapter introduces the proposed method, i.e. *LSH blocking with semantic constraints* in details. First, an overview of the workflow of it is provided. Next the naive method, i.e. LSH blocking without semantic constraints is introduced specifically. Then details of semantic constraints are provided and at the end are the combination of LSH blocking with semantic constraints.

3.1 Overview

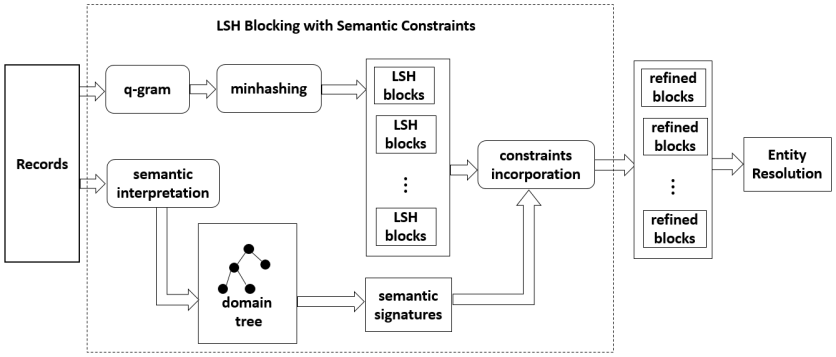


Figure 3.1: Overview of LSH blocking with semantic constraints

Figure 3.1 presents a standard workflow of LSH blocking with semantic con-

straints. As we can see, first, records from datasets are processed in five steps:

1. Q-gram method converts records into sets.
2. Records are blocked on textual similarity by LSH.
3. Semantic interpretations are applied to records.
4. Semantic signatures of records are generated according to a domain tree.
5. Results of LSH blocking are refined by semantic signatures.

In the following sections, details about the related works are provided.

3.2 Locality Sensitive Hashing

3.2.1 Jaccard Similarity and Jaccard Distance

On a textual level, we need to at first choose a measurement of similarity. *Jaccard similarity* [5] is a measurement of the similarity of two sets. It is defined by

$$s_J = \frac{S_1 \cap S_2}{S_1 \cup S_2} \tag{3.1}$$

where S_1 and S_2 is two arbitrary sets and s is their Jaccard similarity.

Example 3.1: In Figure 3.1 we see two sets S_1 and S_2. The number of the elements in their intersection is 4 and the number of elements that appear in S_1 or S_2 is 9. Thus, $s_J(S_1, S_2) = 4/9$ Jaccard distance is defined by

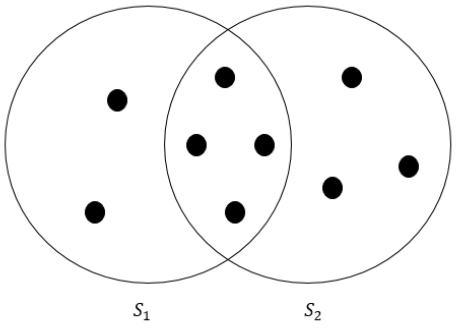


Figure 3.2: Two sets with Jaccard similarity of 4/9

$$d_J = 1 - s_J \tag{3.2}$$

In the following, we will discuss records in terms of their n-gram sets. We can calculate the similarity of two records using their n-gram sets in terms of Jaccard similarity.

Example 3.2: Consider two records chosen from some attributes from Cora dataset, and the values in "Journal" attribute are "Presence" and "Science". We use these two strings to compute the similarity with $q = 2$. Then,

$$s_J(Presence, Science) = \frac{\{Pr, re, es, se, en, nc, ce\} \cap \{Sc, ci, ie, en, nc, ce\}}{\{Pr, re, es, se, en, nc, ce\} \cup \{Sc, ci, ie, en, nc, ce\}} = \frac{3}{10}$$

3.2.2 Minhashing

When calculating similarities of large-size sets of n-grams, a problem arises that it is impractical to store all the information. If we have millions of records, it will be a disaster in space to store all of them. We want to replace large sets by much smaller representations. Such representations should have a property that, they should be comparable to estimate Jaccard similarity of two records. It may impossible for them to give the exact similarity of two records that they represent, but they can make the estimation close to the real value. Such representations are called *signatures*, which will be introduced in details in the following.

- **Characteristic Matrix**
  Characteristic matrix [19] is a representation of sets in a Matrix. The rows of the matrix correspond to the elements of the *universal set*, which contains all the grams from all of the records to be compared. The columns of the matrix correspond to the records. If a record in column c contains an element in row r, then the value of (r, c) is 1. Otherwise the value is 0.

  Example 3.4: Let $S_1 = Presence$, $S_2 = Science$, then table 3.1 presents a characteristic matrix with respect to S_1 and S_2.

Element	S_1	S_2
Pr	1	0
re	1	0
es	1	0
se	1	0
en	1	1
nc	1	1
ce	1	1
Sc	0	1
ci	0	1
ie	0	1

Table 3.1: Characteristic matrix with respect to three records

- **Minhashing**
  In practice, a characteristic matrix usually have more 0's than 1's, thus can not be considered as a efficient way to store the records. We only need to store the positions where 1's appears to save the space.
  The signatures mentioned above are composed of a big number of values, each of which is called a "*minhash* [19]". A minhash of a record is generated from the representing column of the characteristic matrix. Specifically, we randomly permutate the order of the rows of the characteristic matrix, then the minhash value of any column is the number of the first row, in the new order, in which the record has a "1".

  Example 3.5: Table 3.2 is a permutation of the rows of Table 3.1. For S_1, the first row in which the value is 1 is row 2, thus $h(S_1) = 2$ where $h(S_1)$ represent the minhash value of S_1. Similarly, $h(S_2) = 1$

Element	S_1	S_2
Sc	0	1
re	1	0
ci	0	1
se	1	0
en	1	1
ie	0	1
es	1	0
nc	1	1
ce	1	1
Pr	1	0

Table 3.2: A permutation of the rows of the matrix presented in Table 3.1

- **Minhash Signatures**
  If applying n random permutations to the original universal row order, a record S presented by a column will get n minhash values. Consider each of the random permutation as a minhash function, then $h_1(S), ..., h_n(S)$ are minhash values. The vector that consists of these minhash values is a minhash signature of this record. We then construct a *minhash signature matrix* [19] from the characteristic matrix using n minhash functions, in which the ith column is the minhash signature of the record represented by the ith column in the characteristic matrix.

  In literature, a minhash function usually only permutates the rows but still keeps the original row indexes. In my project, to generate a minhash value, a minhash function hashes the original row numbers to distinct values rather than keep the original indexes of the rows. This is fine because the rows are actually randomly

ordered after the hash. This random process is to ensure that our estimate is probabilistically close enough to the exact Jaccard similarity. Note that all of this is based on the following theory

- The probability a minhash function produces the same value for two records equals the Jaccard similarity of them [19].

Based on this, we can simply estimate the Jaccard similarity of two records by compute the ratio of the number of the same minhash values to the number of the minhash functions.

Example 3.5: Table 3.3 presents a signature matrix of S_1, S_2 and S_3.

Minhash functions	S_1	S_2
h_1	2	1
h_2	4	6
h_3	5	5

Table 3.3: Signature matrix with respect to characteristic matrix in Table 3.1

From Table 3.3, we can get minhash signatures for the three records, $[2, 4, 5]$ and $[1, 6, 5]$ respectively.

3.2.3 Locality Sensitive Hashing

When comparing a large number of records, it is still not efficient enough to generate the result by pairwise comparisons, considering that given n records, we need to do $\binom{n}{2}$ computations. Unfortunately, we can not find a way to reduce any computation if our objective is to get the similarity of every pair. However, we could focus our attention on the pairs which are likely to be similar, which LSH can achieve. In the following, detailed techniques about LSH is introduced.

- **Locality-sensitive Functions**
  Let us use $h(S_1) = h(S_2)$ to indicate that S_1 and S_2 are likely to be similar, and $d(S_1, S_2)$ to denote the distance between S_1 and S_2, where S_1 and S_2 are two arbitrary records. As we discussed, there are couple of distance measures between two records. We use d_1 and d_2 to denote two distances in terms of some distance measure, where $d_1 < d_2$. Then a family of functions is *(d_1,d_2,p_1,p_2)-sensitive* [19] if the functions of this family:

  1. If $d(x, y) \leq d_1$, then the probability that h(x)=h(y) is at least p_1.
  2. If $d(x, y) \geq d_2$, then the probability that h(x)=h(y) is at most p_2.

We can use minhash functions and Jaccard distance to find a family locality-sensitive functions. Using the concepts we discussed, we can conclude that the family of minhash functions is a $(d_1, d_2, 1-d_1, 1-d_2)$-sensitive family.

- **Locality Sensitive Hashing for Minhash Signatures**
  Based on minhashing and Jaccard distance, locality sensitive hashing hashes the pairs to several buckets. Let us pay attention back to minhashing. After we get minhash signature matrix for records, LSH divides columns into l bands consisting of k rows for each. For each band, we take vectors of k minhash values, and records with the same vector in the same band will get grouped into the same bucket. Then any pair within the same bucket is considered as a *candidatepair*. Therefore a dataset can be blocked well with LSH. After blocking with LSH, matching process compares only candidate pairs and does not bother to care about the rest.

  In LSH process, we hope to avoid that two records that are not similar still are hashed into the same bucket. But there is still a chance of this happening, which is called a *false positive* [11]. Similarly, two records that are actually similar are hashed into the same bucket is called a *true positive* [11]. We also hope that all truly similar pairs are hashed to the same bucket. But if not, such kind of pair is called a *false negative* [11].

  The algorithm for LSH blocking is shown in Algorithm 1.

Algorithm 1 Locality Sensitive Hashing for Minhash Functions

Input: R is the record set
S_{min} is a list of arrays standing for minhash signatures of records
l is the number of bands
k is the number of rows in each band
Output: B is a list of sets of blocks

```
1: B ← {} //set B to empty
2: for i = 0 to l − 1 do
3:     B_i ←buckets of records grouped by hash(S_min[i ∗ k, (i + 1) ∗ k])
4:     B.append(B_i) //add all the blocks to B
5: end for
6: return B
```

- **Analysis of Collision Probability**
  Collision probability is the chance that two records can be considered as a candidate pair, i.e. is grouped into the same bucket at least in one band in the LSH process. Here we still use k to denote the number of signatures in each band, and l to denote the number of bands. In addition, we use s to denote the Jaccard similarity of two records. As discussed above, s is also equal to the probability that

two signatures of these two records are the same in any one row of the signature matrix. Therefore, we can calculate that

1. the probability the signatures of the two records are the same in all rows in a certain band is s^k
2. the probability the signatures of the two records are different in any one row in a certain band is $1 - s^k$
3. the probability the signatures of the two records are different in any one row in all bands is $(1 - s^k)^l$
4. the probability the signatures of the two records are the same in all rows in at least one band is $1 - (1 - s^k)^l$

Thus, the collision probability is $1 - (1 - s^k)^l$. This function is in the form of $S - curve$, as shown in Figure 3.2.
Under $k = 3$ and $l = 10$, values of p changes w.r.t. s as shown in Table 3.4.

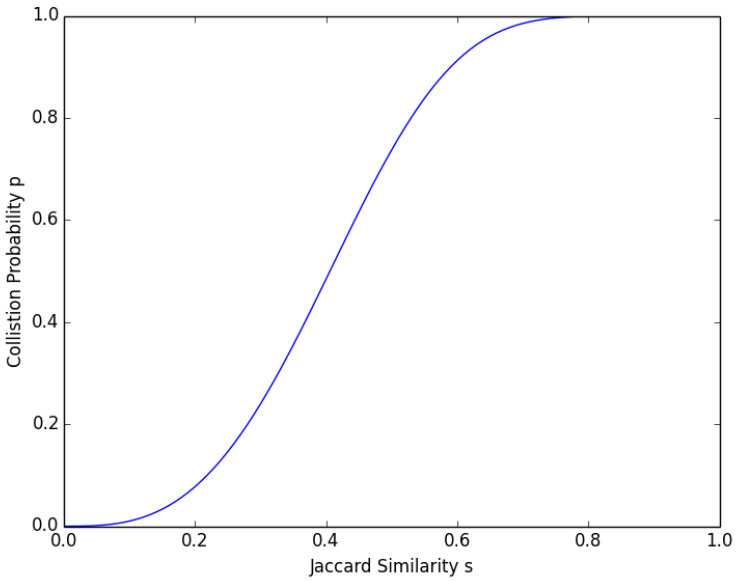


Figure 3.3: $S - curve$

s	p
0.1	0.010
0.2	0.077
0.3	0.239
0.4	0.484
0.5	0.737
0.6	0.912
0.7	0.985
0.8	0.999

Table 3.4: Values of s and p given that $k = 3$ and $l = 10$

In Table 3.3, we see that a pair with the $s = 0.1$ only have 0.01 chance to be considered as a candidate pair, while a pair with the $s = 0.6$ has a chance of 0.912. The parameter k decides how difficult a candidate pair can be considered "likely to be similar" in a certain band, which in a way controls the number of false positives, and l decides how many chances a pair has to take the "test" controlled by k, which in contrast, controls the number of false negatives. The tuning of LSH parameters, i.e. q, k and l is difficult, I will talk about how I choose these parameters in details in the next chapter.

3.3 Semantic Constraints

Semantic constraints are some semantic rules according to which some candidate pairs generated by LSH, will be filter out to make the results more accurate. In this part, several semantical concepts are introduced. A domain tree contains categorical information from a specific domain, and semantic interpretations are used to generate a set of categories for each data record. According to the domain tree and semantic interpretations, semantic signatures of records can be generated.

3.3.1 Domain Tree

A *domain tree* is a tree structure that extracts the information of categories from domain knowledge, independent on specific datasets. From a domain tree, we can draw logical rules between different objects in this domain. The leaves are the ultimate categories which can not be divided further.

Example 3.5: Figure 3.3 shows the domain tree of research outputs. From this tree, we can draw two kinds of rules.

- If A and B are at the same node, which is not a leave, then they have a chance to be in the same category.

Figure 3.4: Domain tree for research output

- If A and B share the same parent, but not at the same node, then they must not be in the same category.

Thus based on these rules, we can reach a decision whether two records are semantically similar or not in this domain.

3.3.2 Semantic Interpretations

When comparing two records according to a domain tree, we may at first get these two records from some datasets. These datasets are not necessary to be the same one, therefore we need to interpret the categorical information from different datasets to the same domain tree.
A *semantic interpretation* of a dataset is a set of functions that can extract the information from the dataset to solely categories.

Example 3.6: Table 3.6 is a semantic interpretation of the dataset presented in Table 3.5 w.r.t. the domain tree shown in Figure 3.3. This semantic interpretation refer records to nodes in of the domain tree according to values of three attributes of records, as shown in Table 3.7. This set of rules are complete, i.e. it covers all the records in dataset.

Id	Journal	Booktitle	Institution
123	Presence	NIPS 3	
124	Science		
125	Machine Learning		
126	Vision Research		University of Michigan
127			

Table 3.5: An example dataset

Journal	Institution	Booktitle	Semantic interpretation
not null	not null	not null	journal, conference, report, thesis
not null	not null	null	journal, report, thesis
not null	null	not null	journal, conference
not null	null	null	journal
null	not null	not null	conference, report, thesis
null	not null	null	report, thesis
null	null	not null	conference
null	null	null	publication

Table 3.6: Semantic Interpretation

Id	Semantic interpretation
123	journal, report, thesis
124	journal
125	journal
126	journal, conference
127	publication

Table 3.7: Result of semantic interpretation

3.3.3 Semantic Signatures

A *semantic signature* is used to indicate what categories a record may be in. After semantic interpretation, each record may refer to one or more nodes then the nodes are projected down to the leaves. Then an n-length binary sequence is used to denote the semantic signature of a record, where n is the number of leaves of the domain tree. The ith 1 in the sequence denote a record may be in the category w.r.t. the ith leaf, while the ith 0 in the sequence denote a record may not.

Example 3.6: Table 3.5 shows the semantic signatures of the records in Table 3.4 w.r.t. the domain tree shown in Figure 3.3, and here $n = 6$.

Id	Semantic signature
123	1 1 0 0 0 0
124	1 0 0 0 0 0
125	1 0 0 0 0 0
126	1 0 0 1 1 0
127	1 1 1 1 1 0

Table 3.8: Semantic signatures of the dataset in Table 3.4

3.4 LSH Blocking with Semantic Constraints

Now we get both the minhash signatures and the semantic signatures, and the question left is to combine these two together to decide whether two records are likely to be similar.

Please recap from 3.2.3, within each band, the algorithm hashes the records into different buckets by comparing k rows of minhash signatures of all the records. Within a bucket, there are records which are likely to be similar to each other based on textual similarity.

Now we incorporate the semantic constraints into this process. If the k rows of signatures of two records are identical, then we evaluate whether these two records obey the semantic constraints according to their semantic signatures. Their are couple of ways to set up the semantic constraints, in my project, I apply two methods as shown below.

1. Randomly choose an index of the signature, and compare the indexed signatures of two records. Take it for m rounds, if both of the indexed signature are 1 in all the rounds, then these two records obey semantically constraints.

2. Randomly choose an index of the signature, and compare the indexed signatures of two records. Take it for m rounds, if both of the indexed signature are 1 in at least one round, then these two records obey semantically constraints.

3.4.1 Logical Conjunction

The first method is in a way a method of logical conjunction, while the second method is a method of logical disjunction. Obviously, the first method is more strict to judge whether two records are semantically similar, thus can control the number of false positives.

After we combine these two part together, the related probability we discussed in 3.2.2 change. We use s_1 to denote the probability that two minhash signatures of these two records are the same in any one row of the minhash signature matrix, and s_2 to denote the probability that any indexed signatures of these two records are the same. For the method of logical conjunction, we have

1. the probability the signatures of the two records are the same in all rows in a certain band is ${s_1}^k {s_2}^m$

2. the probability the signatures of the two records are different in any one row in a certain band is $1 - {s_1}^k {s_2}^m$

3. the probability the signatures of the two records are different in any one row in all bands is $(1 - {s_1}^k {s_2}^m)^l$

4. the probability the signatures of the two records are the same in all rows in at least one band is $1 - (1 - {s_1}^k {s_2}^m)^l$

Thus collision probability is $1 - (1 - {s_1}^k {s_2}^m)^l$ for this case.

The algorithm is shown in Algorithm 2.

Algorithm 2 LSH with Logical Conjunction Constraints

Input: R is the record set
B is the result of LSH blocking
S_{sem} is a list of arrays standing for semantic signatures of records
l is the number of bands
k is the number of rows in each bands
m is the number of rules
Output: BS is the result of refined blocks

```
1: BS ← {} //set BS to empty
2: for i = 0 to l − 1 do
3:    BS_i ← {} //set the set of blocks of band i to empty
4:    idx ← {} //a list of randomly generated indexes
5:    for j to m do
6:       idx ← idx ∪ {Random(0, m − 1)} //add an index to list
7:    end for
8:    BS_i ← filter(B_i, idx, S_sem) //filter out all the records with values of 0, chosen
      by random indexes from semantic signatures
9:    BS ← BS ∪ {BS_i} //add all the blocks to BS
10: end for
11: return BS
```

3.4.2 Logical Disjunction

In the contrast, *logical disjunction* in a way multiple the chance of two records being considered as semantically similar by m, thus can control the number of false negatives. This is similar to how k and l control the precision and recall.

As for the collision probability for this case, we have

1. the probability the signatures of the two records are the same in all rows in a certain band is ${s_1}^k(1 - (1 - s_2)^m)$

2. the probability the signatures of the two records are different in any one row in a certain band is $1 - {s_1}^k(1 - (1 - s_2)^m)$

3. the probability the signatures of the two records are different in any one row in all bands is $(1 - {s_1}^k(1 - (1 - s_2)^m))^l$

4. the probability the signatures of the two records are the same in all rows in at least one band is $1 - (1 - {s_1}^k(1 - (1 - s_2)^m))^l$

Thus the collision probability is $1 - (1 - {s_1}^k(1 - (1 - s_2)^m))^l$ for this case.

The algorithm is shown in Algorithm 3.

Algorithm 3 LSH with Logical Disjunction Constraints

Input: R is the record set
B is the result of LSH blocking
S_{sem} is a list of arrays standing for semantic signatures of records
l is the number of bands
k is the number of rows in each bands
m is the number of rules
Output: BS is the result of refined blocks

1: $BS \leftarrow \{\}$ //set BS to empty
2: **for** $i = 0$ to $l - 1$ **do**
3: $BS_i \leftarrow \{\}$ //set the set of blocks of band i to empty
4: $idx \leftarrow \{\}$ //a list of randomly generated indexes
5: **for** j to m **do**
6: $idx \leftarrow idx \cup \{Random(0, m - 1)\}$ //add an index to list
7: **end for**
8: $BS_i.addAll(break(B_i, idx, S_{sem}))$ //break LSH blocking blocks into smaller blocks indexed by random indexes, according to the values 1 chosen by random indexes from semantic signatures
9: $BS \leftarrow BS \cup \{BS_i\}$ //add all the blocks to BS
10: **end for**
11: **return** BS

Chapter 4

Evaluation

This chapter focuses on the evaluation of the proposed blocking method. It includes the experiment design, experimental results and discussion. In 4.1,experiment design is specifically introduced to shows how the experiment will be performed. It mainly includes test dataset, metrics, compared methods and experimental environment. Then, the comparisons of the proposed method and the traditional method without semantic constrains are discussed. Finally, a brief summary is presented at the end.

4.1 Experiment Design

4.1.1 Dataset

In my project, I used the Publications table from the Cora dataset to evaluate my framework. This table contains 1879 records about machine learning publications. Table 4.1 describes of the attributes it contains.

Attribute	Description	Note
id	Record ID	primary key
pid	Publiction ID	trivial and mostly missing
authors	Author names	
title	Publication title	
publisher	Publisher	
address	Address of publisher	
note	Additional note	
pdate	Date of publication	
journal	Journal name	
volume	Volume number	
pages	Pages	
tech	Technical report title	
institution	Institution name	
booktitle	Book title	
booktile	Mistaken spell of booktitle	trivial and mostly missing
editor	Editor	
year	Publication year	
type	Publication type	
month	Publication month	
clusterid	Cluster Id	

Table 4.1: Publications table from Cora dataset

This dataset is quite dirty, with many duplications, missing values and wrong spellings.

4.1.2 Metrics

In literature, usually completeness, ratio of reduction and F-measure are used to test the performance of a blocking method. In this project, pair precision and pair redundancy are also used, due to the concern to check the precision and redundancy of records in the proposed method. Though the ratio of reduction and F-Measure are also used in this project, we care the most about pair completeness and pair precision as well as pair redundancy. The metrics are introduced in the following.

- **Pairs Completeness**
  Pairs completeness (PC) [13] can be expressed using the formula:

$$PC = \frac{TP_B}{TP_G} \tag{4.1}$$

where TP_B is the number of true positives in the blocks generated by LSH and TP_G is the number of pairs that should be matched according to the ground truth. The PC matric here reflects the notion of recall in my project.

- **Pairs Precision**
  Pairs precision (PP) can be expressed using the formula:

$$PP = \frac{TP_B}{P_{NR}} \tag{4.2}$$

  where P_{NR} here is the number of pairs in the blocks without considering the redundancies.
  The PP matric here reflects the notion of precision in my project.

- **Pairs Redundancy**
  Pairs redundancy (PR) can be expressed using the formula:

$$PR = 1 - \frac{P_{NR}}{P_R} \tag{4.3}$$

  where P_R here is the number of pairs in the blocks counting the redundancies.
  The PR matric here reflects the ratio of redundancies of my blocking framework.

- **Reduction Ratio**
  Reduction ratio (RR) [13] can be expressed using the formula:

$$RR = 1 - \frac{P_{NR}}{P_A} \tag{4.4}$$

  where P_A here is the number of all pairs in the original dataset. The RR matric here reflects the extent to which my blocking method reduces the number of pair-wise matches.

- **F-Measure**
  F-measure (FM) [13] can be expressed using the formula:

$$FM = \frac{2 * RR * PC}{RR + PC} \tag{4.5}$$

  What we want is to achieve is to reduce pair-wise comparisons as much as possible without much completeness loss. However there must be a trade-off between two of them. The FM matric here is to succinctly express this trade-off.

4.1.3 Compared Methods

To see whether semantic constraints can bring benefits to blocking process, the naive method is used in this experiment to compare with our proposed method. The details of these methods that need to be compared are listed below:

- **LSH Blocking with Semantic Constraints(LBS):** The proposed method, i.e. LSH blocking with semantic constraints.
- **LSH Blocking without Semantic Constraints(LB):** The naive blocking method, i.e. LSH blocking without semantic constraints.

4.1.4 Experimental Environment

This section shows the experimental environment which includes: programming language, IDE, relevant tools and experimental environment that are planned to use. They are shown in the table below:

Experimental environment	Semantic signature
Programming Language	Java 1.7
Programming IDE	Netbeans 8.0
Relevant tool	Postgres Database

Table 4.2: Experimental environment

4.2 Experimental Results

4.2.1 Parameter Setting

There are a number of parameters involved in my project:

1. size of q-gram
2. k and l

These parameters are a key part in my project. In the following, details about tuning of these parameters are provided subsequently.

- **Size of Q-gram**
  Above all, we should choose the size of q-gram. On the textual level, I choose the combination of the values of two attributes, authors and titles, to evaluate the textual similarity between two records. Thus this size is dependant on the length of this combined string. Figure 4.1 shows the similarity distribution when q equals to different values. Specifically, I compared similarities of all similar pairs, and rounded them to keep one decimal place. Amounts of pairs of respective similarities are counted, then I computed ratios of respective amounts to the

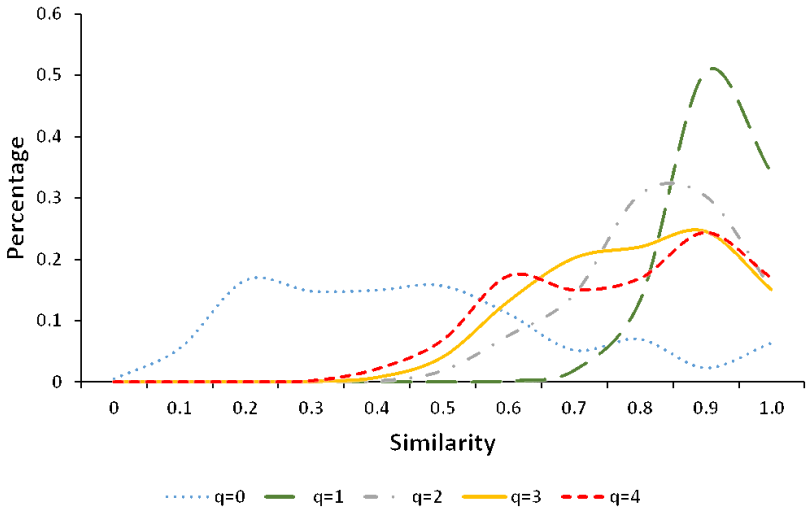


Figure 4.1: Similarity distribution of similar pairs

amounts of all similar pairs. The so-called similar pairs here are the pairs that should be blocked together, which can be drawn from the ground true of the original dataset.
From Figure 4.1 we can observe that when $q = 0$, all the similarities are larger than 0, which indicates nothing at all. While too small q may be to expensive for computation, work in [16] shows $q = 4$ is a good choice for European languages.

- **Value of k and l**
  After we choose 4 as the q-gram size, we can draw from Figure 4.1 that most of the similar pairs has a similarity of larger than 0.3, thus 0.3 can be as a similarity threshold, based on which pairs are considered. In my project, I want to guarantee that a pair of records with a similarity lager than 0.3 will be blocked together with a bounded probability p. In this way, we actually ensure an acceptable recall, which is essential, because this is exactly what a blocking process is subject to. Now we have the fixed value of similarity s and collision probability p, then the number of bands can be expressed in terms of fixed k, s, and p

$$l = \frac{\ln(1-p)}{\ln(1-s^k)} \tag{4.6}$$

  Here (4.6) is based on the textual LSH collision probability without considering the constraints. It is because, there is another parameter, i.e. the number of semantic signatures m, to be tuned, which is difficult. So we just do it in a simple and explicit way, i.e. each round we calculate the value of l based on the fixed k, s and p, then we select a proper m independent on these values. Table 4.3 shows the l's w.r.t. k from 1 to 6.

k	l
1	2
2	6
3	19
4	63
5	210
6	701

Table 4.3: Values of l subject to different k

4.2.2 Comparison of LBS to LB

Before any comparisons, results for a baseline is tested first. Considering that, in this project, the semantic signatures are short and randomly chosen in every band, therefore this randomness may interfere with the effectiveness of my framework inevitably. Thus this project needs to establish a baseline to judge whether my approach works well above all.

Maximum Probability Constraint (MPC) is such a semantic constraint that applies the logical disjunction rule to signatures one by one, instead of randomly choosing them, with $m = 6$. In this way, each pair will get the largest chance to be considered as a candidate pair when incorporating semantic constraints. Table 4.4 is the evaluations with MPC.

k	l	PC	PP	PR	RR	FM
1	2	0.9049	0.5232	0.4070	0.9367	0.9205
2	6	0.9345	0.6256	0.5628	0.9453	0.9399
3	19	0.9345	0.6256	0.5628	0.9453	0.9399
4	63	0.9902	0.7546	0.8230	0.9520	0.9707
5	210	0.9908	0.6383	0.8593	0.9432	0.9664
6	701	0.9960	0.7113	0.8953	0.9488	0.9718

Table 4.4: Blocking evaluation with MPC

From Table 4.4, we get a boundary of blocking performance with semantic constraints. For example, under the setting $k = 4$ and $l = 63$, no matter what kinds of semantic constraints we apply, ER blocking will get a PC value at most 0.9345 and get a PP value at least 0.7546.

- **Pair Completeness**

  Figure 4.2 presents the PC values of blocking without constraints and blocking with constraints. Here the constraints is set to logical disjunction method with 3 semantic signature values. For the results of other semantic constraints, details

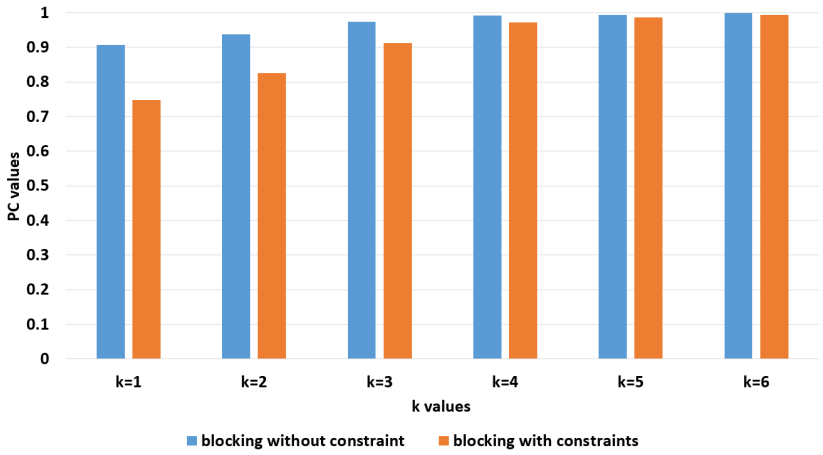


Figure 4.2: PC values of blocking without constraints and with constraints

are provided in Appendix.

From Figure 4.2, we observe that except for the cases $k = 1$ and $k = 2$, all the PC values of the naive method without constraints are pretty high. This is reasonable we calculate the value of l based on this condition. For $k = 1$ and $k = 2$, the numbers of hash functions are too small to approximate the Jaccard Similarity with minhash functions from a statistical perspective, thus the PC values are somehow relatively abnormal.

We can also observe that PC values of the new method is acceptable, because with the growth of the value of k, PC values grows rapidly and when k equals to 6, it is approximately the same as that of the naive method.

- **Pair Precision**

Figure 4.3 presents the PP values of blocking without constraints and blocking with constraints, and still the constraints is set to logical disjunction method with 3 semantic signature values. From Figure 4.3, we can observe that the new method with constraints improves PP values a great deal.

Please note that for the naive method without constraints, with k increasing from 1 to 4, PP gets higher, but when $k = 4$ and 5 it gets lower again, which is abnormal. Explanations are provided in the following.

In my project, according to the definition of PP, PP is the ratio TP to the

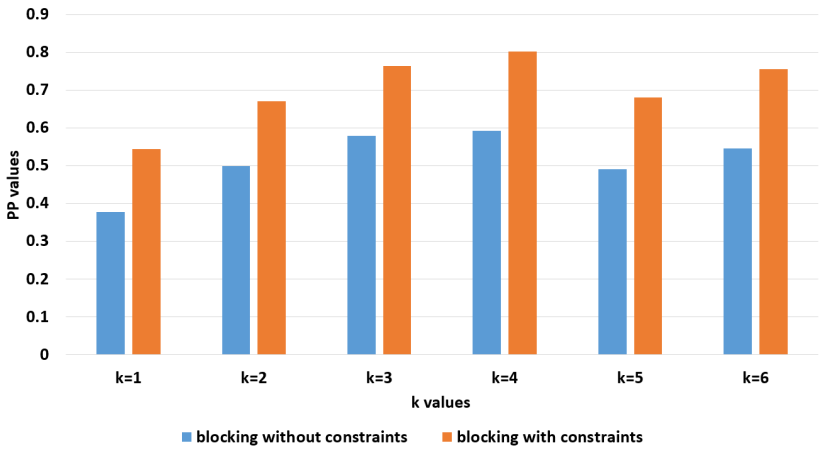


Figure 4.3: PP values of blocking without constraints and with constraints

sum of the TP and FP, i.e.

$$PP = \frac{TP}{TP + FP} \tag{4.7}$$

Let us use FP_G to denote the number of pairs that should not be matched according to the ground truth. Then we have

$$TP_G + FP_G = P_A \tag{4.8}$$

Let us use $P(C_1)$ to denote the probability that a pair which should be matched together will get matched by LSH, $P(C_2)$ to denote the probability that a pair which should not be matched together will get matched by LSH. Then we have

$$P(C_1) = \sum_{S_1} P(S_1)P(C_1|S_1) \tag{4.9}$$

where S_1 is the similarity of pairs that should be matched together, and

$$P(C_1|S_1) = 1 - (1 - (S_1)^k)^l \tag{4.10}$$

which is exactly the collision probability. And we can derive $P(S_1)$, the similarity distribution from the original dataset. Similarly, we can also have

$$P(C_2) = \sum_{S_2} P(S_2)P(C_2|S_2) \tag{4.11}$$

where S_2 is the similarity of pairs that should not be matched together, and

$$P(C_2|S_2) = 1 - (1 - (S_2)^k)^l \tag{4.12}$$

Given $P(C_1)$ and $P(C_2)$, TP and FP will follows the *binomial distribution* [1], i.e. $TP \sim B(TP_G, P(C_1))$ and $FP \sim B(FP_G, P(C_2))$. Note that TP and FP here is the number of these two kinds of pairs.

Due to the property of binomial distribution, we can get

$$E(TP) = TP_G P(C_1) \tag{4.13}$$

and

$$E(FP) = FP_G P(C_2) \tag{4.14}$$

We can approximate PP as

$$\begin{aligned} PP &\approx \frac{E[TP]}{E[TP] + E[FP]} \\ &= 1/(1 + \frac{E[FP]}{E[TP]}) \end{aligned} \tag{4.15}$$

Based on this, let us look at this problem again. We can observe that when k equals from 1 to 4, the PC has big growths in ratio, which means that from k=1 to $k = 4$, with the growth of k, more pairs that should be matched are actually matched by LSH, while in the meantime, more pairs that should not be matched also get matched based on textual similarity. When k =5, PC is very close to 100%, which means TP hardly grows any more. But FP still grows a lot regardless of TP. This is exactly why the PP values get lower when k is larger than 4.

The new method with constraints also ameliorates the circumstances we just discussed above.

- **Pair Redundancy**
  PR values are also tested as shown in Figure 4.4. Pair redundancy is really important because it reflects two kinds of costs in LSH blocking process. At first, consider that within a LSH blocking, between different bands, LSH blocking may generate many duplicate candidate pairs. If we have computed the similarity of a pair, we do not need to compute this again in the following bands. On the other hand, it will still be needed to check whether the similarity of a pair has computed. Therefore one cost of two is the cost to compute pair similarity and the other one is to check whether a pair is a redundancy. However, we can not simply say whether a higher or a lower PR value is better, because it depends on the size of blocks and the number of blocks. More details will be discussed in 4.2.4.

  To simplify, just consider the cost to check redundancy, and Figure 4.4 shows that the proposed method decreases this cost slightly. This is reasonable because when adding constraints, the chance that two records which has been blocked together will be blocked together again in another band decreases, especially for false positives.

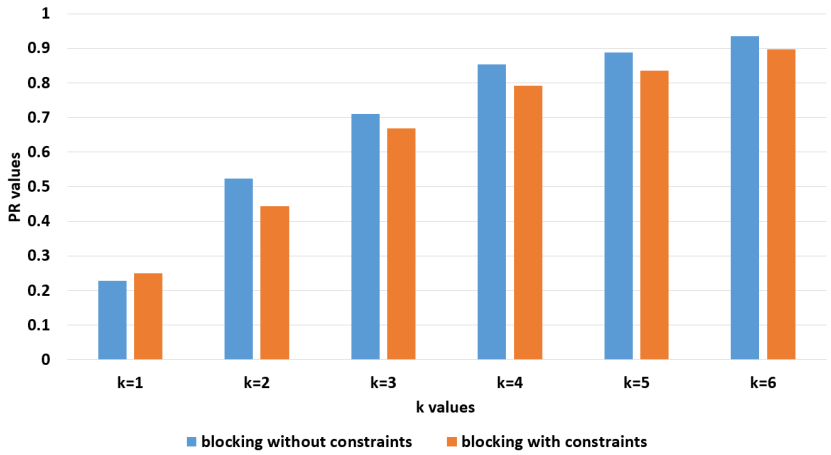


Figure 4.4: PR values of blocking without constraints and with constraints

4.2.3 Comparison of LBS with Different Constraints

Comparisons between LBS methods with different constraints are also conducted. Here the comparison is for logical disjunction with 3 signatures and logical conjunction with 2 signatures. Due to the randomness of minhash functions and semantic constraints, metrics measured in Table 4.5 and 4.6 are average values of ten computations, followed by standard deviations.

k	l	$\overline{PC}$	σ_{PC}	$\overline{PP}$	σ_{PP}	$\overline{PR}$	σ_{PR}	$\overline{RR}$	σ_{RR}	$\overline{FM}$	σ_{FM}
1	2	0.7471	0.1155	0.5442	0.0528	0.2497	0.0846	0.9497	0.0066	0.8308	0.0740
2	6	0.8258	0.0700	0.6705	0.0311	0.4443	0.0628	0.9548	0.0048	0.8838	0.0383
3	19	0.9132	0.0597	0.7629	0.0195	0.6691	0.0327	0.9562	0.0025	0.9331	0.0313
4	63	0.9730	0.0146	0.8017	0.0100	0.7923	0.0297	0.9556	0.0008	0.9642	0.0069
5	210	0.9864	0.0035	0.6805	0.0157	0.8360	0.0104	0.9469	0.0012	0.9662	0.0018
6	701	0.9931	0.0011	0.7550	0.0188	0.8970	0.0213	0.9518	0.0012	0.9720	0.0007

Table 4.5: Blocking evaluation with logical disjunction of 3 signature

k	l	$\overline{PC}$	σ_{PC}	$\overline{PP}$	σ_{PP}	$\overline{PR}$	σ_{PR}	$\overline{RR}$	σ_{RR}	$\overline{FM}$	σ_{FM}
1	2	0.1820	0.1367	0.6733	0.1359	0.1645	0.1419	0.9892	0.0077	0.2861	0.1803
2	6	0.3121	0.1446	0.7438	0.1288	0.1784	0.0804	0.9840	0.0075	0.4560	0.1541
3	19	0.5126	0.1534	0.8351	0.0614	0.4470	0.0670	0.9775	0.0065	0.6574	0.1407
4	63	0.6511	0.0990	0.8743	0.0296	0.5796	0.0344	0.9727	0.0046	0.7753	0.0713
5	210	0.8209	0.0562	0.7851	0.0542	0.6468	0.0540	0.9615	0.0037	0.8846	0.0318
6	701	0.9225	0.0218	0.8504	0.0171	0.8258	0.0229	0.9603	0.0010	0.9409	0.0110

Table 4.6: Blocking evaluation with logical conjunction of 2 signature

With the growth of k, deviations are getting smaller, which means the results get more stable and consistent.

- **Pair Completeness**
  Figure 4.5 shows the comparison between PC values of blocking with two different semantic constraints corresponding to above two tables. Blocking with logical conjunction constraints has lower PC values, compared to blocking with logical disjunction constraints.

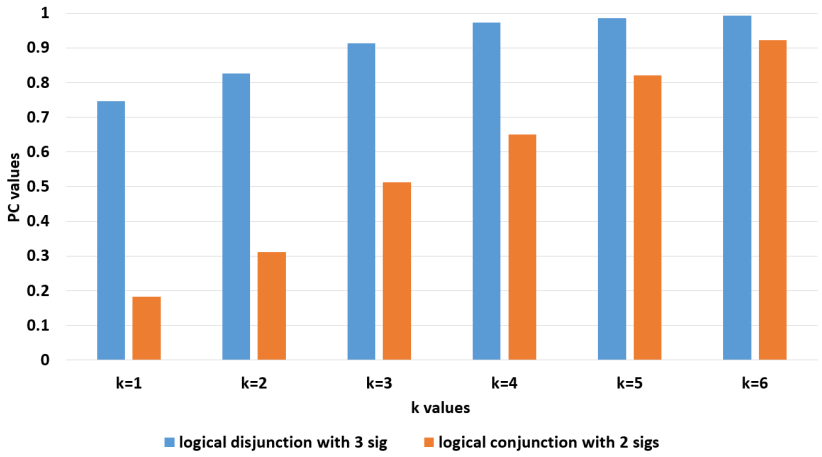


Figure 4.5: PC values of blocking with different constraints

- **Pair Precision**
  Figure 4.6 shows the comparison between PP values of blocking with two different semantic constraints corresponding to above two tables. Blocking with logical conjunction constraints has higher PC values, compared to blocking with logical disjunction constraints.

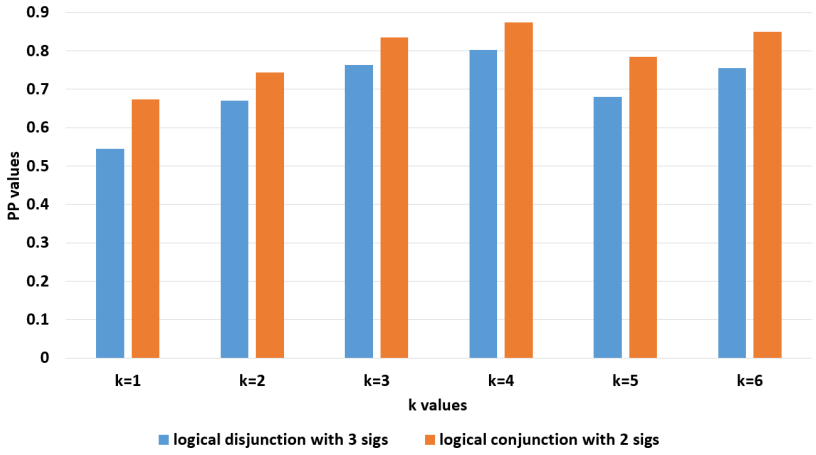


Figure 4.6: PP values of blocking with different constraints

4.2.4 Block Size Distribution

Block size distribution is also tested, which is relevant to pair redundancies. Figure 4.7 shows numbers of blocks in different settings of semantic constraints. Figure 4.8 shows block size distributions for different k values.

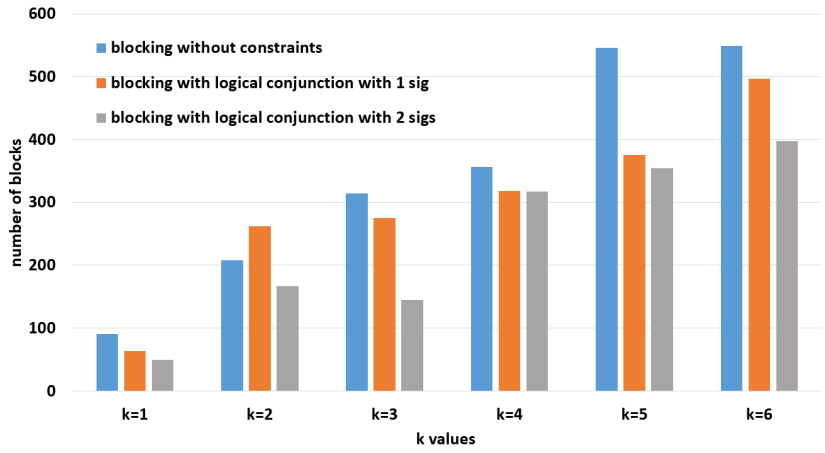


Figure 4.7: Number of blocks

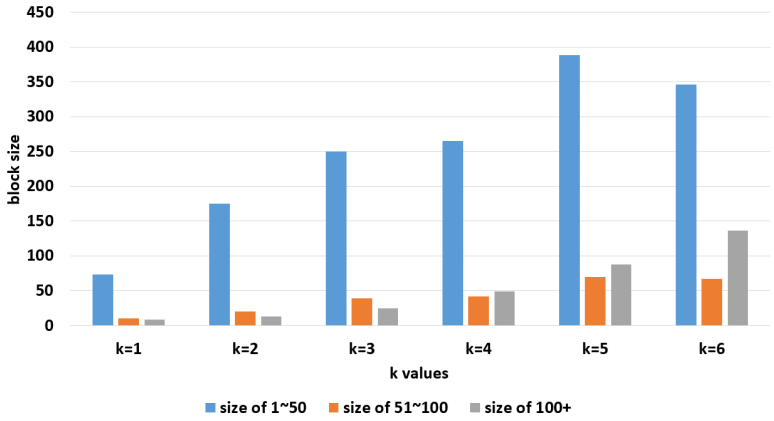


Figure 4.8: Block size distribution

From Figure 4.7, without constraints, the larger k is, the more blocks LSH will generate. This is because the bigger k is, the harder two records can be blocked together in a band, though a bigger l provides more chances for them to be considered. Therefore, big blocks break into small ones, which can be drawn from Figure 4.8, and in this way, the number of blocks get larger. It is a naive analysis, things will get much more complicated when I incorporate constraints.

Note that it is abnormal that when $k = 1, 3, 4, 5, 6$, numbers of blocks all get smaller with constraints, but when $k = 2$, the amount gets bigger. Let us use B_C to denote the blocks with constraints, B to denote the blocks without constraints and $S(B)$ to denote the number of blocks in B. Then there are two circumstances which may happen in LSH blocking.

1. If a block generated by textual LSH is illegal subject to the semantic constraints, it will be broken into smaller blocks.

2. If a block generated by textual LSH is illegal subject to the semantic constraints, it will get removed directly.

Here that a block is illegal means in this block, one or more pairs break the semantic constraints. To clearly explain these, I will use small-size datasets as examples.

Example 4.1 Say the original dataset contains five records (a,b,c,d,e), then $B = ((a, b), (a, c), (b, d), (c, e))$. Taking semantic constraints into consideration, the pair (a,c) breaks semantic constraints, thus $B_C = ((a, b), (b, d), (c, e))$. From B, the pair (a,c) got removed, thus $S(B_C) < S(B)$, which corresponds to the first circumstance shown above.

Example 4.2 Still with the same dataset that contains five records, and $B = ((a, b), (c, d, e))$. Taking semantic constraints into consideration, the pair (c,d) breaks semantic constraints, thus $B_C = ((a, b), (c, d), (c, e))$. From B , the pair (c,d,e) was broken into smaller blocks (c,d),(c,e), thus $S(B_C) > S(B)$, which corresponds to the second circumstance shown above.

Comparing these two examples, we can draw that whether an illegal block subject to the constraints can be broken into smaller blocks matters. When a block is checked to be illegal subject to the constraints, it will be broken into smaller blocks first, and if there already exists the same blocks as them, then such blocks will be removed. The second circumstance often happens when the blocks have a large number of redundancies. What I mean redundancies here are those pairs appear in more than one block. Therefore, information of redundancy is a key part to explain this, no matter that of the true positives or of the false positives.

4.3 Summary

This chapter evaluated the pair completeness, precision and redundancy of LSH blocking with semantic constraints compared to LSH blocking without constraints. Firstly, experiment design is discussed which covers tested dataset, metrics, etc. For datasets, Cora dataset is used. Regarding evaluation metrics, pair completeness, precision and redundancy are decided to be used to measure the performance of the proposed method and the naive method.

In the second part, parameter setting for q, k and l are investigated. In the comparison part, new method(LBS) is firstly compared with the naive method (LB) on Cora dataset for PC values. The results show the completeness of proposed method performs better with the growth of k value. When k equals to 6, it is almost the same as that of the naive method. Then another experiment focus on pair precision is conducted. The outcomes show that the proposed approach improves pair precision a great deal. The results of this part show proposed method is more effective than the naive one. Regarding PR values, the outcome shows the new method decreases redundancies.

Chapter 5

Conclusion

The main purpose of this project is to incorporate semantic constraints into LSH blocking process to achieve scalable, high performance entity resolution. In this work, we propose an approximate blocking approach with semantic constraints. First, LSH blocking is adopted to group the records based on the textual similarity, and then semantic constraints are used to refined the generated LSH blocks.

The experiments demonstrate the effectiveness of the proposed method. The challenge is to improves PP values while keeping an acceptable PC values and to optimize the process of pairwise comparison. The proposed method, i.e. LSH blocking with semantic constraints improves PP values significantly without big loss of PC. Also for scalability of entity resolution, my framework improves RR, i.e. reduces more pairwise comparisons.

From experiments, some limitations of LSH blocking are found:

- It is hard to determine the control parameters, i.e k and l. Though in this report, l values can be determined by (4.6), it is often more related to the specific datasets.
- It will be a problem to handle excessive space issues. To ensure good quality of blocking result, a large number of hash functions are often required.

Future Work

The future work would lie on three points:

1. Evaluate LBS over more than one datasets.
2. Evaluate LBS for runtime and memory usage.
3. Investigate deeper about a choice of semantic rules.

Appendices

Blocking Performance

k	l	PC	PP	PR	RR	FM
1	2	0.9074	0.3769	0.2282	0.9119	0.9096
2	6	0.9370	0.4993	0.5233	0.9313	0.9341
3	19	0.9739	0.5784	0.7101	0.9384	0.9558
4	63	0.9927	0.5927	0.8532	0.9387	0.9650
5	210	0.9933	0.4899	0.8880	0.9258	0.9584
6	701	0.9985	0.5451	0.9355	0.9330	0.9646

Table 1: Blocking evaluation without constraints

k	l	$\overline{PC}$	σ_{PC}	$\overline{PP}$	σ_{PP}	$\overline{PR}$	σ_{PR}	$\overline{RR}$	σ_{RR}	$\overline{FM}$	σ_{FM}
1	2	0.3797	0.1883	0.5724	0.0838	0.1030	0.1061	0.9765	0.0096	0.5199	0.1846
2	6	0.6400	0.1105	0.7109	0.0399	0.2657	0.0757	0.9669	0.0062	0.7644	0.0773
3	19	0.7868	0.0652	0.8129	0.0353	0.5600	0.0330	0.9645	0.0029	0.8651	0.0376
4	63	0.9193	0.0299	0.8454	0.0180	0.7310	0.0313	0.9602	0.0014	0.9390	0.0149
5	210	0.9704	0.0093	0.7342	0.0284	0.7860	0.0158	0.9516	0.0017	0.9609	0.0048
6	701	0.9854	0.0048	0.8016	0.0140	0.8811	0.0184	0.9550	0.0007	0.9699	0.0025

Table 2: Blocking evaluation with logical conjunction of 1 signature

k	l	$\overline{PC}$	σ_{PC}	$\overline{PP}$	σ_{PP}	$\overline{PR}$	σ_{PR}	$\overline{RR}$	σ_{RR}	$\overline{FM}$	σ_{FM}
1	2	0.1820	0.1367	0.6733	0.1359	0.1645	0.1419	0.9892	0.0077	0.2861	0.1803
2	6	0.3121	0.1446	0.7438	0.1288	0.1784	0.0804	0.9840	0.0075	0.4560	0.1541
3	19	0.5126	0.1534	0.8351	0.0614	0.4470	0.0670	0.9775	0.0065	0.6574	0.1407
4	63	0.6511	0.0990	0.8743	0.0296	0.5796	0.0344	0.9727	0.0046	0.7753	0.0713
5	210	0.8209	0.0562	0.7851	0.0542	0.6468	0.0540	0.9615	0.0037	0.8846	0.0318
6	701	0.9225	0.0218	0.8504	0.0171	0.8258	0.0229	0.9603	0.0010	0.9409	0.0110

Table 3: Blocking evaluation with logical conjunction of 2 signature

k	l	$\overline{PC}$	σ_{PC}	$\overline{PP}$	σ_{PP}	$\overline{PR}$	σ_{PR}	$\overline{RR}$	σ_{RR}	$\overline{FM}$	σ_{FM}
1	2	0.4513	0.1321	0.5674	0.0778	0.0817	0.0715	0.9711	0.0069	0.6043	0.1219
2	6	0.5980	0.1333	0.6957	0.0719	0.2837	0.0732	0.9688	0.0050	0.7302	0.1049
3	19	0.7875	0.0732	0.7990	0.0282	0.5452	0.0396	0.9639	0.0032	0.8649	0.0428
4	63	0.9219	0.0266	0.8450	0.0157	0.7346	0.0553	0.9600	0.0014	0.9404	0.0133
5	210	0.9531	0.0300	0.7384	0.0221	0.8014	0.0241	0.9527	0.0021	0.9527	0.0145
6	701	0.9819	0.0073	0.7950	0.0108	0.8778	0.0239	0.9548	0.0006	0.9681	0.0035

Table 4: Blocking evaluation with logical disjunction of 1 signature

k	l	$\overline{PC}$	σ_{PC}	$\overline{PP}$	σ_{PP}	$\overline{PR}$	σ_{PR}	$\overline{RR}$	σ_{RR}	$\overline{FM}$	σ_{FM}
1	2	0.6138	0.1668	0.5623	0.0607	0.2078	0.0583	0.9605	0.0090	0.7330	0.1413
2	6	0.7397	0.1107	0.6692	0.0599	0.3912	0.0509	0.9598	0.0037	0.8298	0.0825
3	19	0.8976	0.0578	0.7757	0.0257	0.6152	0.0416	0.9577	0.0021	0.9256	0.0315
4	63	0.9628	0.0196	0.8160	0.0197	0.7818	0.0315	0.9568	0.0015	0.9597	0.0094
5	210	0.9833	0.0029	0.6926	0.0175	0.8117	0.0202	0.9480	0.0013	0.9653	0.0013
6	701	0.9900	0.0029	0.7618	0.0136	0.8990	0.0269	0.9524	0.0008	0.9708	0.0015

Table 5: Blocking evaluation with logical disjunction of 2 signature

k	l	$\overline{PC}$	σ_{PC}	$\overline{PP}$	σ_{PP}	$\overline{PR}$	σ_{PR}	$\overline{RR}$	σ_{RR}	$\overline{FM}$	σ_{FM}
1	2	0.7471	0.1155	0.5442	0.0528	0.2497	0.0846	0.9497	0.0066	0.8308	0.0740
2	6	0.8258	0.0700	0.6705	0.0311	0.4443	0.0628	0.9548	0.0048	0.8838	0.0383
3	19	0.9132	0.0597	0.7629	0.0195	0.6691	0.0327	0.9562	0.0025	0.9331	0.0313
4	63	0.9730	0.0146	0.8017	0.0100	0.7923	0.0297	0.9556	0.0008	0.9642	0.0069
5	210	0.9864	0.0035	0.6805	0.0157	0.8360	0.0104	0.9469	0.0012	0.9662	0.0018
6	701	0.9931	0.0011	0.7550	0.0188	0.8970	0.0213	0.9518	0.0012	0.9720	0.0007

Table 6: Blocking evaluation with logical disjunction of 3 signature

k	l	$\overline{PC}$	σ_{PC}	$\overline{PP}$	σ_{PP}	$\overline{PR}$	σ_{PR}	$\overline{RR}$	σ_{RR}	$\overline{FM}$	σ_{FM}
1	2	0.6225	0.1615	0.5380	0.0648	0.2789	0.0543	0.9575	0.0114	0.7397	0.1340
2	6	0.8406	0.0620	0.6454	0.0451	0.4429	0.0446	0.9523	0.0027	0.8916	0.0358
3	19	0.9371	0.0212	0.7569	0.0151	0.6813	0.0398	0.9547	0.0011	0.9457	0.0105
4	63	0.9827	0.0096	0.7836	0.0073	0.7937	0.0361	0.9541	0.0005	0.9682	0.0046
5	210	0.9884	0.0020	0.6692	0.0149	0.8484	0.0111	0.9459	0.0012	0.9667	0.0012
6	701	0.9940	0.0012	0.7439	0.0140	0.8964	0.0186	0.9511	0.0009	0.9721	0.0005

Table 7: Blocking evaluation with logical disjunction of 4 signature

Number of Blocks

k	l	Number of blocks
1	2	91
2	6	208
3	19	314
4	63	356
5	210	546
6	701	549

Table 8: Number of blocks without constraints

		Number of blocks	
k	l	m=1	m=2
1	2	64	49
2	6	262	167
3	19	275	145
4	63	318	317
5	210	376	354
6	701	497	398

Table 9: Number of blocks with logical conjunction constraints

Block Size Distriubtion

Block size	Frequency					
	$k = 1$,$l = 2$	$k = 2$,$l = 6$	$k = 3$,$l = 9$	$k = 4$,$l = 63$	$k = 5, l = 210$	$k = 6, l = 701$
1∼50	73	175	250	265	388	346
51∼100	10	20	39	42	70	67
100+	8	13	25	49	88	136

Table 10: Block size distributions

Bibliography

[1] A. H. Ang and W. H. Tang. Probability concepts in engineering. *Planning*, 1(4):1–3, 2004.

[2] M. Bawa, T. Condie, and P. Ganesan. Lsh forest: self-tuning indexes for similarity search. In *WWW*, pages 651–660, 2005.

[3] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

[4] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[5] M. Chandrasekharan and R. Rajagopalan. Groupabil1ty: an analysis of the properties of binary data matrices for group technology. *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, 27(6):1035–1052, 1989.

[6] P. Christen. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer, 2012.

[7] P. Christen. Data matching. data-centric systems and applications, 2012.

[8] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[9] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611, 2007.

[10] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[11] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[12] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[13] M. Kejriwal and D. P. Miranker. An unsupervised algorithm for learning blocking schemes. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 340–349. IEEE, 2013.

[14] H.-s. Kim and D. Lee. Harra: fast iterative hashed record linkage for large-scale data collections. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 525–536. ACM, 2010.

[15] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[16] P. Mcnamee and J. Mayfield. Character n-gram tokenization for european language text retrieval. *Information Retrieval*, 7(1-2):73–97, 2004.

[17] B. Okner. Data matching and merging: an overview. In *Annals of Economic and Social Measurement, Volume 3, number 2*, pages 49–54. NBER, 1974.

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

[19] A. Rajaraman and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2012.

[20] S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. Entity resolution with iterative blocking. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 219–232. ACM, 2009.

[21] S. Yan, D. Lee, M.-Y. Kan, and L. C. Giles. Adaptive sorted neighborhood methods for efficient record linkage. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 185–194. ACM, 2007.