Entity Resolution with Active Learning

Jingyu Shao

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

October 2021

I certify that the thesis is my own original work, except where otherwise indicated.

Jingyu Shao
10 October 2021

Dedicated to my dear parents and my dear wife.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor, Dr. Qing Wang for her continuous support of my research and PhD study. Her guidance helped me all the time of my research and writing the thesis. Her enthusiasm on research teaches me to be respectful on my work; her motivation encourages me to find new research problems and go deeper into them; her patience helps me to keep clam when suffering from complex issues; her immense knowledge guides me to avoid unnecessary mistakes. There are countless cases that she gave me insightful comments during my research life to improve the quality of my works at different stages of the research problems. I could not have imagined having a better supervisor for my PhD study.

Besides Dr. Qing Wang, I would also like to thank my co-supervisors, Prof. Peter Christen and Dr. Kerry Taylor, for their constructive feedback and guidance on my research and annual progress report.

I would like to thank Dr. Yu Lin, Fangbing Liu, Asiri Wijesinghe and Prof. Erhard Rahm, who helped me in different research problems and finalized them into publications. I can still remember Yu always shared his experience with me, he is wisdom, can always propose new ways of thinking, try different possible solutions during the discussion.

I'm always thinking to be fortunate enough to have many nice colleagues with me in the Research School of Computer Science. The warm-hearted CS administration team members Christie, Jasmine, Melissa and so on helped me a lot on dealing with some bureaucratic processes. Dr. Josh Milthorpe and Dr. Jeffrey Fisher helped me on dealing with academical teaching. Graeme, Andrew from Helpdesk provided me IT support and solved my problems at the firt time.

Last but not the least, I would like to show my deep gratitude to my family members: my wife and my parents for their unconditional support. They helped me walk through the most difficult time in my PhD career.

Publications

Primary Publications

Contributions from work presented in this thesis have been published across multiple peer-reviewed conferences and journals. A list of publications in reverse chronological order is given below:

- J. Shao, Q. Wang, A. Wijesinghe and E. Rahm. ERGAN: Generative Adversarial Networks for Entity Resolution. The 20th IEEE International Conference on Data Mining (ICDM), 2020.

  The concepts and related terms of this generative-based approach, theoretical analysis, algorithm implementation and evaluations of this paper are presented in Chapter 7.

- J. Shao, Q. Wang and F. Liu. Learning To Sample: an Active Learning Framework. The 19th IEEE International Conference on Data Mining (ICDM), 2019.

  The preliminaries of this learning-based active learning framework, theoretical analysis, algorithm implementation and evaluations of this paper are presented in Chapter 6.

- J. Shao, Q. Wang and Y. Lin. Skyblocking for Entity Resolution. Information Systems (IS), Elsevier, 2019.

  The problem definition of this scheme skyline framework, theoretical analysis, algorithm implementation and evaluations of this paper are presented in Chapter 5.

- J. Shao and Q. Wang. Active Blocking Scheme Learning for Entity Resolution. The 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2018.

  The problem definition of this blocking scheme learning approach, theoretical analysis, algorithm implementation and evaluations of this paper are presented in Chapter 4.

Secondary Publications

One more paper is published as a join work with other collaborators. However, contributions from this publication, even though being closely related is not presented in this thesis. This paper is:

- Q. Bui-Nguyen, Q. Wang, J. Shao, and D. Vatsalan. Repairing of Record Linkage: Turing Errors into Insight. The 22nd International Conference on Extending Database Technology (EDBT), 2019.

Abstract

Entity Resolution refers to the process of identifying records which represent the same real-world entity from one or more datasets. In the big data era, large numbers of entities need to be resolved, which leads to several key challenges, especially for learning-based ER approaches: (1) With the number of records increasing, the computational complexity of the algorithm grows exponentially. (2) Quite a number of samples are necessary for training, but only a limited number of labels are available, especially when the training samples are highly imbalanced.

Blocking technique helps to improve the time efficiency by grouping potentially matched records into the same block. Thus to address the above two challenges, in this thesis, we first introduce a novel blocking scheme learning approach based on active learning techniques. With a limited label budget, our approach can learn a blocking scheme to generate high quality blocks. Two strategies called active sampling and active branching are proposed to select samples and generate blocking schemes efficiently. Additionally, a skyblocking framework is proposed as an extension, which aims to learn scheme skylines. In this framework, each blocking scheme is mapped as a point to a multi-dimensional scheme space where each blocking measure represents one dimension. A scheme skyline contains blocking schemes that are not dominated by any other blocking schemes in the scheme space. We develop three scheme skyline learning algorithms for efficiently learning scheme skylines under a given number of blocking measures and within a label budget limit.

While blocks are well established, we further develop the *Learning To Sample* approach to deal with the second challenge, i.e. training a learning-based active learning model with a small number of labeled samples. This approach has two key components: a sampling model and a boosting model, which can mutually learn from each other in iterations to improve the performance of each other. Within this framework, the sampling model incorporates uncertainty sampling and diversity sampling into a unified process for optimization, enabling us to actively select the most representative and informative samples based on an optimized integration of uncertainty and diversity. On the contrary of training with a limited number of samples, a powerful machine learning model may be overfitting by remembering all the sample features. Inspired by recent advances of generative adversarial network (GAN), in this paper, we propose a novel deep learning method, called ERGAN, to address the challenge. ERGAN consists of two key components: a label generator and a discriminator which are optimized alternatively through adversarial learning. To alleviate the issues of overfitting and highly imbalanced distribution, we design two novel modules for diversity and propagation, which can greatly improve the model generalization power. We theoretically prove that ERGAN can overcome the model collapse and convergence problems in the original GAN. We also conduct extensive experiments to empirically verify the labeling and learning efficiency of ERGAN.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Considering when you are scheduling a holiday trip, you can always find the cheapest flight tickets on the website from different agents even for the same flight. When you are searching for a paper or textbook with its title, website links from different sources may be provided. Such things happen everyday to make our life easier, which is associated with a term: *entity*.

An entity normally refers to a thing that is distinct and unique existence in the real-world, such as a person, a business company or even a place to go. In database systems, an entity can be stored as a record with descriptions of the entity, which can be the color of a car, the venue of a publication or the address of a person. Table 1.1 shows a dataset example of smartphones, iPhone X and Samsung Galaxy S9 Plus with two colors, respectively.

Entity resolution techniques have been studied for about half a century since Fellegi et al. first proposed the theoretical concepts of probabilistic record linkage in 1969 [47]. Entity Resolution (ER) is of great importance in many applications, such as matching product records from different online stores for customers, detecting people's financial status for national security, or analyzing health conditions from different medical organizations [26; 50]. For example, if a national census agency wants to obtain the population growth of its country in a time period, ER is necessary to detect whether records from different time points refer to the same person, no matter whether he or she changed name (e.g. due to marriage), postal address and so on. In the Big-data era, large organizations deal with millions of records every day to discover useful knowledge for decision making, and even a person can deal with thousands of records if he or she types in one key word on a searching website. Normally, these websites integrate records

Table 1.1: **An smartphone dataset referring to four entities.**

Record Number	Name	Color	Storage	Screen Size
r_1	Apple iPhone X	Silver	64GB	5.8"
r_2	Apple iPhone X	Space Gray	256GB	5.8"
r_3	iPhone X	Space Gray	256GB	5.8"
r_4	Apple iPhone X	Space Gray	256GB	-
r_5	Samsung Galaxy S9 Plus	Midnight Black	64GB	6.2"
r_6	Galaxy S9 Plus	Black	64GB	-
r_7	Galaxy S9 Plus	Coral Blue	-	-

Figure 1.1: A general process of ER. One or more datasets after pre-processing (e.g. data cleaning) are used as input, and clusters of records referring to the same real-world entities are generated as output.

of multiple datasets from different data sources. For example, different online-shopping websites such as *ebay* and *amazon* may provide different descriptions on the same product, on the other hand, different product records may have similar descriptions as well. Such cases may confuse the customers so that they may fail to tell one from another and thus make a poor decision. How to process a large number of records and find out the real-world entity they refer to is becoming more and more important.

Conceptually, *Entity Resolution*, which is also called *Record Linkage* [62; 70], *Deduplication* [27] or *Data Matching* [26], refers to the process of identifying records which represent the same real-world entity from one or more datasets [147]. Traditionally, the entity resolution process contains four steps: Blocking, Comparison, Classification and Clustering as shown in Fig 1.1.

- Blocking: Given one or several sets of records, blocking aims to group all the records into different blocks, so that only potentially matched records within a block need to be resolved referring to the same entity. For example, schema-agnostic blocking technique considers a subset of attribute values, and records sharing the same values will be in the same block. Blocking is a very important process in ER in that it can minimize the number of comparisons without affecting the accuracy of resolution significantly.

- Comparison: Given records within a block, a pre-defined function is used to compare record values pairwise by measuring the similarity of values, and it returns a vector for each record pair indicating the degree of similarity of the two records. Standard comparison functions includes n-gram Jaccard similarity, edit distance and so on [26].

- Classification: In this step, the feature vectors indicating the similarity of each record pair are categorized as matches and non-matches as their labels, indicating whether the corresponding records referring to the same real world entity or not. For example, a machine learning-based classifier is normally trained based on labeled feature vectors (as samples) and used to predict other vectors' labels.

- Clustering: The final step comes to identify all the records referring to the same entity into the same cluster, so that all records within one cluster are matches and referring to a single entity. This step also helps to identify those records hardly to be directly identified as matches corresponding to the same entity. E.g., A and B is match, B and C is match but A and C is non-match [26].

1.2 Challenges in Entity Resolution

A large number of approaches have been proposed in recent years to deal with aforementioned issues. These approaches to classify the records can be either supervised or unsupervised. Most of unsupervised approaches classify record pairs based on their similarities without any label information, i.e., the ground truth (matches or non-matches) of record pairs [84], thus the predicted classes may not be promised. For example, it is hard to define a proper similarity threshold to determine whether a record pair is a match, and similar records may refer to different entities. Supervised approaches, on the other hand, use prior knowledge on part of the record pairs in a dataset to train a classifier, and apply this classifier to the rest of the dataset for label prediction. However, the ground truth is expensive and sometimes even unable to achieve [26], due to two reasons: (1) the labels are normally provided by the domain experts, which may contain human mistakes; (2) considering to compare 1,000 records in a dataset pairwise, there will be in-total 499,500 labels; with the increasing size of a dataset, obtaining all the labels becomes harder and more expensive.

Designing a model which can solve ER tasks with a limited number of ground truth labels is not easy. There are three key challenges involved.

- **Scalability:** In ER tasks, in order to identify all the records referring to the same entity, traditionally, every two records need to be compared as a pair. Given two datasets D_1 and D_2, the number of pairwise comparisons is thus $|D_1| \times |D_2|$. This is not efficient for large datasets. As a result, blocking techniques can be applied to group potentially matched records into the same block, such that only records in the same block will be compared in details using comparison functions. With the application of blocking, large numbers of potentially non-match record pairs will not be compared. However, it is still a problem to find good blocking schemes w.r.t. various criteria, such as pair completeness (similar to recall) or pair quality (similar to precision) [11].

- **Label Cost:** When the number of record pairs are large, generating their labels (matches or non-matches) is very expensive. Various approaches were proposed to improve the time efficiency for blocking and classification in recent years [58; 62; 2], but few of them referred to the label efficiency for supervised approaches, i.e., how to train a good classifier with only a small number of labeled samples. Additionally, the class imbalance problem exists in ER tasks, i.e., more non-matches than matches for the record pairs of a dataset (the majority of pairs correspond to non-matches) [149]. Traditional classification methods need a large percentage of training set with random selected samples, to guarantee its training performance and allow more matches to be selected. If we can alleviate the class imbalance problem, the number of samples and labels we need will decrease. Thus one challenge is how to obtain a (nearly) balanced training set with a limitation number of labels?

- **Quality:** Using supervised learning-based classification models is an efficient and promising way to solve ER tasks and categorize record pairs into matches and non-matches when the labeled training samples are sufficient. However, if the labeled samples are limited, the quality of the performance can not be guaranteed. The models are so powerful that they can remember the features of all the samples, where the overfitting problem

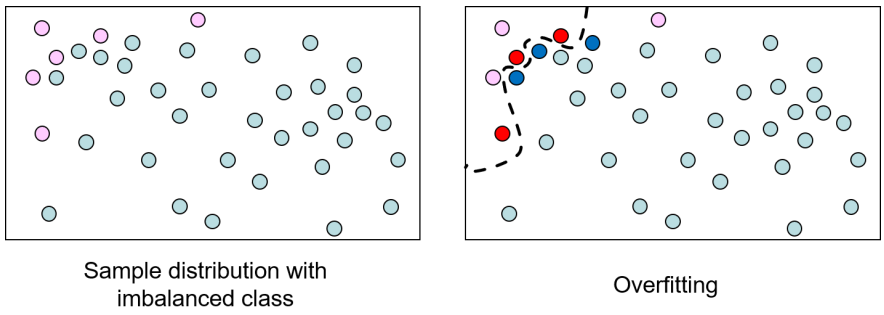

Figure 1.2: An example of the overfitting problem in ER classification. The dash line indicates how the model classifies samples.

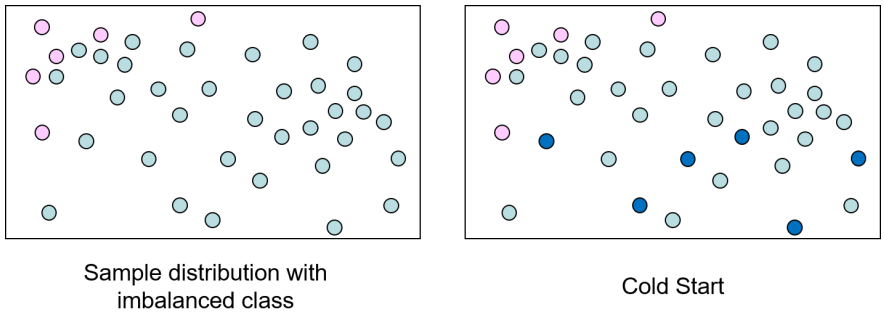

Figure 1.3: An example of the cold start problem in ER classification. Samples in solid colors are labeled for training, which are all from the majority class, i.e., samples in blue.

occurs. The natural of ER tasks aggravates this problem since the samples are highly imbalanced, i.e., most of the samples are from the majority class. Thus how to train a model with high performance under a limited number of labeled samples is still a problem in solving ER tasks.

These challenges are not standing alone. With a limited number of training samples, the overfitting problem is more likely to occur, where the label cost and the quality challenges are related. Due to the class imbalance problem, there are far more non-matches than matches in the training set, which will affect the performance of a machine learning model. Additionally, when the samples are highly imbalanced, in the worst case, all the samples are from the majority class and thus the cold start problem occurs. These problems are shown in Fig. 1.2 and Fig. 1.3, where the red and blue points refer to matches and non-matches, respectively. Specifically, in Fig. 1.2, a powerful model can easily distinguish the training samples with different labels, even they are quite similar to each other in the sample space. However, such models lose their generalization. All these problems will aggravate the difficulty of training a classification model in solving ER tasks.

1.3 Research Objectives

The aforementioned challenges can be summarized into two questions: (1) How to resolve entities efficiently? (2) How to resolve entities accurately? This thesis addresses these challenges from two perspectives in ER: (1) blocking and (2) classification.

1.3.1 Blocking Objectives

In ER, with the size of a dataset growing, the similarity computation time for records increases quadratically. To eliminate the unnecessary computation and improve the time efficiency, blocking techniques are widely applied by grouping potentially matched records into blocks, and restricting the comparison only occurring between records in the same block. For example, given a dataset D, the total number of record pairs to be compared is $\frac{|D|*(|D|-1)}{2}$ (i.e., each record is paired with all other records in D). Using blocking technique can reduce the number of compared record pairs to no more than $\frac{m*(m-1)}{2} * n$, where m is the number of records in the largest block and n is the number of blocks.

In past years, a good number of techniques have been proposed for blocking [43; 146; 50; 121; 122; 135], such as sorted neighborhood based blocking [43], locality-sensitive hashing (LSH) based blocking [146], clustering based blocking [35; 50], graph based blocking [121; 122], and scheme based blocking [11; 135; 84; 108; 85; 17]. Among these techniques, using blocking schemes is an efficient and declarative way to generate blocks. Intuitively, a blocking scheme takes records from a dataset as input, and groups the records using a logical combination of blocking predicates, where each blocking predicate specifies an attribute and its corresponding function. Thus, using a blocking scheme helps the user to understand how the blocks are generated.

Learning a blocking scheme is the process of deciding which attributes are chosen for blocking, what the corresponding functions are used to compare values in attributes, and how different attributes and methods are logically combined so that desired blocks can be generated to satisfy the given criterion, e.g. at least how many percentages of matches are supposed to be in the blocks. Compared with blocking techniques that consider data at the instance level [43], for example, split and merge specific records based on their own record values [50], blocking schemes have several advantages: (1) They only require to decide what metadata, such as attributes and the corresponding functions, is needed, rather than what data from individual records is selected; (2) They provide a more human readable description for how attributes and methods are involved in blocking; and (3) They enable more natural and effective interaction for blocking across heterogeneous datasets. A number of blocking approaches have been proposed to learn blocking schemes [11; 84; 108]. They generally fall into two categories: (1) Supervised blocking scheme learning approaches. For example, Michelson and Knoblock presented an algorithm called *BSL* to automatically learn effective blocking schemes [108]; (2) Unsupervised blocking scheme learning approaches [84]. For example, Kejriwal and Miranker proposed an algorithm called *Fisher* which uses record similarity to generate labels for training based on the TF-IDF measure, and a blocking scheme can then be learned from a training set [84]. However, as mentioned above, sufficient labeled samples are necessary for supervised learning approaches; and there is no performance guarantee for unsupervised learning

approaches.

Additionally, ER applications often involve multi-criteria analysis in choosing blocking schemes. More specifically, given a scheme space that contains a large number of possible blocking schemes and a collection of criteria for choosing blocking schemes, such as pair completeness (PC), pair quality (PQ) and reduction ratio (RR) [27], how can we select the *most preferred* blocking scheme? Ideally, a good blocking scheme should yield blocks that minimize the number of record pairs to compare, while still preserving true matches as many as possible, i.e., optimizing all criteria simultaneously. Unfortunately, the criteria for selecting blocking schemes are often competing with each other. For example, PC and PQ are negatively correlated in many applications, as well as RR and PQ [27]. That is to say, a blocking scheme with high PC normally leads to a low PQ, and conversely, a blocking scheme with high PQ may have a low PC. From users' perspective, they may have different preferences on blocking schemes to deal with different applications, which may have specific requirements, and thus to achieve entity resolution results from various perspectives. For example,

- *Crime investigation:* In crime investigation, when individuals are investigated for a crime, it is necessary to identify as many candidates as possible so that the criminal will not be missed out. In this case, blocking schemes with high PC values would be preferred.

- *Medical study:* When studying the medical conditions of patients, we would need to identify patients that exactly correspond to certain medical conditions. In this case, blocking schemes with relatively high PQ values would be desired, because they can help match the patients and the medical conditions so as to diagnose patients that are necessarily included under study.

To effectively learn a blocking scheme that is optimal under one or more criteria, previous work has specified various constraints in the learning process [86]. For example, some approaches [17; 108] aimed to learn a blocking scheme that can maximize both RR and PC of record pairs. Some approaches [11; 84] targeted to find a blocking scheme that can generate blocks with a minimal number of non-matched record pairs. However, setting such constraints perfectly is a challenging task because constraints may vary in different applications and it is often unknown which constraint is appropriate for a specific entity resolution task. If a constraint is set strictly, no blocking scheme can be learned; on the other hand, if a constraint is set loosely, the learned blocking scheme may not be helpful in the task.

Finally, we summarize our research objectives for blocking as follows:

- **Objective 1:** To improve the computational efficiency in ER, we first need to build blocks which contain potentially matched records using blocking schemes. With only a limited number of labels available, we consider to learn optimal blocking schemes w.r.t. a user specified criterion.

- **Objective 2:** A number of scheme-based learning approaches have been proposed, however, a properly pre-defined criterion, such as a threshold of pair completeness, is still necessary for learning, which requires the user to be a domain expert with prior knowledge on the dataset. How to overcome this under a limited number of labels is still a

question. Hence, we target to proposing a framework which can present a set of "optimal" blocking schemes under various criteria.

1.3.2 Classification Objectives

The challenges in ER classification are long standing and researchers are trying their best to develop better solutions. Among these solutions, learning-based ER classifiers have been widely used in the past years. However, due to the quadratic nature of record pair comparison required by ER tasks [24], labeling is costly and time consuming. Additionally, these manual labels are highly imbalanced, and most of them are useless. This raises the difficulty of applying supervised learning methods for ER, where the classifier requires a sufficient number of labeled samples for training, which is infeasible in many real-world applications.

To reduce the labeling effort, alternatively, a number of semi-supervised learning methods have been proposed [87; 149; 136]. Some were proposed based on a low-density separation assumption, i.e., there exists a low-density "boundary" so that instances belonging to different classes can be distinguished [6; 101]. However, such a boundary may not always exist or can be clearly identified, especially when the number of labeled instances is small [71]. Some semi-supervised learning methods have utilized the idea of self-learning, which firstly trains a classifier using labeled instances, and then selects unlabeled instances with predicted labels to train a classifier iteratively [87]. Although promising, these methods often lead to the issue of overfitting when labeled instances in training are limited [125].

In this thesis, we first tackle these challenges using the active learning techniques. A number of approaches have been studied to solve the classification problem in entity resolution tasks with active learning and achieved quite high performance [133; 4; 49]. However, current solutions normally used pre-defined heuristic rules for sample selection and labelling, while these rules need prior knowledge and may change w.r.t. different datasets and machine learning models. For example, uncertainty sampling as one kind of active learning techniques is associated with a probabilistic learning model which is used to infer the uncertainty w.r.t. the probability of whether a sample belonging to a certain class [90; 126]. Considering an SVM (Support Vector Machines) based active learning approach, samples which lie closest to the SVM's dividing hyperplane will be selected [132; 45]. However, using such probability may not be reliable since the classifier itself may not be trusted in some cases: (1) A limited number of training samples or a classifier with high complexity may cause the overfitting problem, hence the predicted probabilities are not reliable; (2) When to deal with multi-class problems, a sample with high uncertainty to one class may be certain to another class [72].

The state-of-the-art active learning framework considering both uncertainty and diversity [157], is not efficient in sampling. That is, uncertainty of samples is measured by entropy, which can not be obtained from the classifier until all the samples are actively selected as the training set. Additionally, considering the diversity of samples, the class of each sample is known as prior information, so that samples can be selected from each class evenly, which is not achievable under a limited number of labels. Furthermore, as we have mentioned, the overfitting problem and the cold start problem may occur while training a classifier under a limited number of imbalanced samples. How to alleviate the overfitting problem and the class imbalance problem with a limited number of training samples is still a problem to be explored.

Recent approaches indicate that solving ER tasks with neural networks can achieve better performance than traditional methods such as Magellan [9; 113]. However, without sufficient training data, a powerful machine learning model may be overfitting by remembering all the features of training samples. In such cases, the learning model can correctly predict the classes of seen samples with high certainty, but fail to predict the classes of unseen samples, thus losing the generalization ability. This challenge is further aggravated when the underlying data distribution is highly imbalanced, which raises the difficulty of applying supervised learning methods for ER. Hence we aim to propose a deep learning-based framework which focuses on tackling the following two challenges that cannot be handled by the existing deep learning-based ER methods: (1) **the overfitting problem**; (2) **the imbalanced class problem**.

Thus, our research objectives for classification are as follows:

- **Objective 3:** To avoid using various pre-defined active learning heuristics for different datasets and machine learning models, we consider to design a novel approach where the active learning strategies can be learned from data. Our approach should also deal with the cold start problem when the number of labeled samples is limited.
- **Objective 4:** Existing powerful machine learning models such as neural networks suffer from the overfitting problem when the training samples are limited or the models are powerful enough, especially when the samples are highly imbalanced. In such cases, the models can remember the features of each sample and lose their generalization ability. We aim to design an approach for ER under a limited number of high imbalanced samples.

1.4 Contributions

In this thesis, we focus on the above objectives for both blocking and classification. Inspired by the success of active learning techniques, this thesis provides a detailed study for blocking and classification with active learning techniques, and shows significant improvements in both label efficiency and accuracy compared with the state-of-the-art approaches. Particularly, this is the first time for blocking scheme learning with active learning techniques.

- **Active blocking scheme learning.** To deal with the first objective, i.e., learning the "optimal" blocking scheme under a limited number of labels, we propose an active learning-based approach. This approach contains two complementary and integrated active learning strategies: (a) *Active sampling strategy* which converts the class imbalance problem into the balanced sampling problem and then selects informative training samples; (b) *Active branching strategy* which determines the extension of candidate blocking schemes by using either a conjunction or disjunction form. Our experimental results show that our approach can efficiently learn a blocking scheme with less samples while still achieving high quality compared with the state-of-the-art baselines. Details of this solution are introduced in Chapter 4.
- **Scheme skyline learning.** While the "optimal" blocking schemes can be learned under a given threshold, it may still suffer from the circumstance that the user has no prior

knowledge on defining a proper threshold. To deal with this issue, we formulate a novel scheme skyline learning problem for entity resolution where only a limited number of labels are available. Solving this problem would lead to generating a range of optimal blocking schemes w.r.t. different blocking criteria and thus enable users to choose their preferred blocking schemes. Three algorithms are proposed, where we both actively select informative samples and develop a scheme extension strategy for efficiently identifying schemes that are possibly on a skyline in order to reduce the search space and label cost used in the learning process. We have evaluated the efficiency and effectiveness of our scheme skyline learning algorithms over five real-world datasets. The experimental results show that our algorithms outperform the baseline approaches in all of the following aspects: label efficiency, blocking quality and learning efficiency. Details of this framework are introduced in Chapter 5.

- **Learning-To-Sample (LTS).** While blocks of records are well established w.r.t. the blocking schemes, the next step is to categorize all the record pairs within one block into matches and non-matches for all blocks. Our objective is to design an active learning strategy regardless of the datasets and the machine learning models for entity resolution, and overcome the cold start problem at the same time. Hence we propose a novel learning-based active learning framework, called *Learning-To-Sample* (LTS). In this framework, two models are designed: a boosting model F and a sampling model G, which can dynamically learn from each other in iterations for improving the performance of each other. Additionally, the sampling model incorporates uncertainty and diversity of samples into a unified process for optimization. This allows us to actively select samples based on the joint impacts of probabilities of being mis-classified by a boosting model and the distribution of samples in the sample space. The experimental results show that our approach significantly outperforms all the baselines under a limited number of labels, and efficiently alleviate the cold start problem, especially when the samples are highly imbalanced. Details of this framework are introduced in Chapter 6.

- **ERGAN.** Although the active learning technique helps to reduce the label cost for ER, it still suffers from the overfitting problem when the model is too powerful. To deal with this objective including the overfitting problem and the class imbalance problem, we propose a semi-supervised approach with generative adversarial nets, namely ERGAN, for entity resolution. ERGAN has two key components: a label generator and a discriminator, which are optimized in an adversarial learning manner. We develop two integrated modules: diversity and propagation modules, for the label generator and the discriminator, respectively, to improve the model generalization ability. We theoretically prove that ERGAN overcomes the model collapse and convergence problems in the original GAN, and we conduct extensive experiments to empirically verify the effectiveness of ERGAN over all the baselines and our ablated models. Details of this framework are introduced in Chapter 7.

1.5 Thesis Outline

We begin by discussing the preliminaries for entity resolution in Chapter 2. Then we review the related work and background for ER, skyline queries, active learning, and machine learning models in Chapter 3. In Chapter 4, we propose an active blocking scheme learning approach which incorporates active learning techniques into the blocking scheme learning process while preserving quality. In Chapter 5, based-on the active scheme learning algorithm, we propose a scheme skyline approach w.r.t. different criteria. We also propose a learning-based active learning approach which selects samples by learning instead of pre-defined rules in Chapter 6. We finally propose a neural network based classification model, which contains a label generator and a discriminator forming an adversarial network to deal with the potentially overfitting problem under powerful models and a limited number of labeled samples in Chapter 7. The thesis finishes with a summary, followed by an outlook to possible extensions in Chapter 8.

Chapter 2

Preliminaries

This chapter presents the notations and the experimental setups used in this thesis.

2.1 Notations

Let R be a dataset consisting of records. Each record $r \in R$ is associated with a set of attributes A, and each attribute $a \in A$ has a domain $Dom(a)$. We use $r.a$ to refer to the value of attribute a in a record r. A *blocking function* $h : Dom(a) \times Dom(a) \rightarrow \{0,1\}$ takes a pair of attribute values from $Dom(a)$ as input and returns a value in $\{0,1\}$ as output. A *blocking predicate* $\langle a, h \rangle$ is a pair of a blocking attribute a and a blocking function h. Given a pair of records $\langle r_i, r_j \rangle$, a blocking predicate $\langle a, h \rangle$ returns *true* if $h(r_i.a, r_j.a) = 1$ and returns *false* if $h(r_i.a, r_j.a) = 0$.

Example 2.1.1. *Consider the dataset example in Table 2.1 and suppose that we have a blocking predicate* $\langle$*Authors, Same-soundex*$\rangle$*, where* Authors *refers to the attribute in the dataset, and* Same-soundex *refers to the blocking function applied for this attribute. In Table 2.1, the record* r_1 *has "Gale" in the attribute Authors, while the records* r_2 *and* r_3 *have "Gaile" in the attribute Authors. The soundex value of both "Gale" and "Gaile" is G4. The other records* r_4 *and* r_5 *have "Johnson" in the attribute Authors and the soundex value of "Johnson" is 75. Hence, if we consider the pair of records* $\langle r_1, r_2 \rangle$*, this blocking predicate returns* true *because Same-soundex*$(Gale, Gaile) = 1$*. However, for the pair of records* $\langle r_1, r_4 \rangle$*, the blocking predicate returns* false *because Same-soundex*$(Gale, Johnson) = 0$*.*

Given a set of blocking predicates P, the *blocking vector* of a record pair $\langle r_i, r_j \rangle$ for blocking is defined as $\mathbf{v} = \langle v_1, v_2, ...v_{|P|} \rangle$, where each v_k $(k = 1, \ldots, |P|)$ is a value of either 1 or 0, describing whether the corresponding blocking predicate in P returns true or false, respectively.

A *(blocking) scheme* s is a disjunction of conjunctions of blocking predicates (i.e. in the disjunctive normal form). Given a blocking scheme s, a *blocking model* can generate a set of pairwise disjoint blocks $B_s = \{b_1, \ldots, b_{|B_s|}\}$, where $b_k \subseteq R$ $(k = 1, \ldots, |B_s|)$, $\bigcup_{1 \leq k \leq |B_s|} b_k = R$ and $\bigwedge_{1 \leq i \neq j \leq |B_s|} b_i \cap b_j = \varnothing$. Moreover, for any two records r_i and r_j in a block $b_k \in B_s$, s must contain a conjunction of block predicates such that $h(r_i.a_k) = h(r_j.a_k)$ holds for each block predicate $\langle a_k, h \rangle$ in this conjunction.

A blocking scheme is called a *n-ary blocking scheme* if it contains n distinct blocking predicates. Given a blocking scheme $s = s_1 \vee s_2 ... \vee s_n$, where each s_i $(i = 1, \ldots, n)$ is a

Table 2.1: A bibliographic dataset example with three attributes: Title, Authors and PublicationYear.

ID	Title	Authors	PublicationYear
r_1	An active learning ...	Gale	2003
r_2	An active learning ...	Gaile	2003
r_3	Entity resolution for ...	Gaile	2006
r_4	An active learning ...	Johnson	2003
r_5	Active learning blocking ...	Johnson	2003

conjunction of blocking predicates, we can generate a set of pairwise disjoint blocks $B_s = \{b_1, \ldots, b_{|B_s|}\}$, where $b_k \subseteq R$ $(k = 1, \ldots, |B_s|)$, $\bigcup_{1 \leq k \leq |B_s|} b_k = R$ and $\bigwedge_{1 \leq i \neq j \leq |B_s|} b_i \cap b_j = \varnothing$. Two records r_i and r_j are placed into the same block if there exists a conjunction of block predicates s_i in the given blocking scheme s such that the feature vector x of $\langle r_i, r_j \rangle$ contains 1 for each blocking predicate in s_i. We say $s(x) = true$ in this case; otherwise, $s(x) = false$.

Example 2.1.2. *Given two blocking schemes* $s_1 = \langle$*Authors, Same-soundex*$\rangle \wedge \langle$*Title, Same-value*$\rangle$ *and* $s_2 = \langle$*Authors, Same-soundex*$\rangle \wedge \langle$ *PublicationYear, Same-value*$\rangle$, $s = s_1 \vee s_2$ *is a 3-ary blocking scheme. This is because* s *contains three distinct blocking predicates, i.e.* $\langle$*Authors, Same-soundex*$\rangle$, $\langle$ *PublicationYear, Same-value*$\rangle$, *and* $\langle$*Title, Same-value*$\rangle$.

By applying the blocking scheme s *on the records in Table 2.1, a set of blocks* $B_s = \{\{r_1, r_2\}, \{r_3\}, \{r_4, r_5\}\}$ *would be generated since* s *returns true for* $\langle r_1, r_2 \rangle$ *as well as* $\langle r_4, r_5 \rangle$.

The *feature vector* of a record pair $\langle r_i, r_j \rangle$ for classification is a tuple $\langle sim(r_i.a_1, r_j.a_1), sim(r_i.a_2, r_j.a_2), ...sim(r_i.a_{|A|}, r_j.a_{|A|}) \rangle$, where each $sim(r_i.a_k, r_j.a_k)$ $(k = 1, \ldots, |A|)$ is a value between 0 and 1, referring to the similarity of the corresponding attribute values, which is measured by similarity functions used in the comparison step of the entity resolution process. A *sample*, denoted as x, is defined as either (1) a blocking vector for blocking tasks or (2) a feature vector for classification tasks. For each sample x, a *human oracle* ζ is used to provide a label $y \in \{M, N\}$. If $y = M$, it indicates that $\langle r_i, r_j \rangle$ refers to the same entity (i.e. *a match*), and analogously, $y = N$ indicates that $\langle r_i, r_j \rangle$ refers to two different entities (i.e. *a non-match*). The human oracle ζ is associated with a budget limit $budget(\zeta) \geq 0$, which indicates the total number of labels ζ can provide. A *training set* $T = (X, Y)$ consists of a set of samples $X = \{x_1, x_2, ..., x_{|T|}\}$ and their labels $Y = \{y_1, y_2, ..., y_{|T|}\}$, where X is the *sample set*.

Example 2.1.3. *By applying the 2-gram Jaccard similarity function on the records* r_1 *and* r_2 *in Table 2.1, the feature vector* $x = \langle 1, 0.4, 1 \rangle$.

A summary of notations used in this thesis is presented in Table 2.6.

2.2 Experimental Setups

We will introduce the datasets and the measures used in this thesis. Additionally, all the algorithms are implemented in Python 2.7.3, and running on a server with 6-core 64-bit Intel Xeon 2.4 GHz CPUs, 128 GBytes of memory.

Table 2.2: **Characteristics of datasets for entity resolution.**

Dataset	Cora	DBLP-Scholar	DBLP-ACM	NCVoter
# Attributes	4	4	4	18
# Records	1,295	2,616 / 64,263	2,616 / 2,294	267,716 / 278,262
# True Matches	17,184	2,360	2,224	6,122,579
# Blocking Predicates	16	16 / 16	16 / 16	72 / 72
Class Imbalance Ratio	1 : 49	1 : 31,440	1 : 1,117	1 : 2,692

Table 2.3: **Attributes of datasets.**

Datasets	Attributes
Cora	authors, title, affiliation, publisher and year
DBLP-Scholar	title, authors, venue and year
DBLP-ACM	
NCVoter	county_id, county_desc, voter_reg_num, voter_status_desc, voter_status_reason_desc, absent_ind, last_name, first_name, midl_name, full_name_rep, full_name_mail, reason_cd, status_cd, house_num, street_name, street_type_cd, res_city_desc and state_cd

2.2.1 Datasets

We will use the following four datasets in the experiments to evaluate the performance of our approaches:

- **Cora** [1] dataset contains bibliographic records of machine learning publications including four attributes.
- **DBLP-Scholar** [1] dataset contains bibliographic records from the DBLP and Google Scholar websites including four attributes.
- **DBLP-ACM** [95] dataset contains bibliographic records from the DBLP and ACM websites including four attributes.
- **North Carolina Voter Registration (NCVoter)** [2] dataset contains real-world voter registration information of people from North Carolina in the USA. Two sets of records including 18 attributes are collected in October 2011 and December 2011 respectively are used in our experiments.

The characteristics of these datasets are summarized in Table 2.2, including the number of attributes, the number of records (/ is used to separate the numbers for two datasets respectively), the number of true matches, the number of blocking predicates used in the experiments

[1] Available from: *http://secondstring.sourceforge.net*
[2] Available from: *http://alt.ncsbe.gov/data/*

Table 2.4: **The notions tp, tn, fp, fn defined in blocking evaluation w.r.t. a blocking scheme s.**

True Label	# Record Pairs	
	In blocks	Out of blocks
Match	$tp(s)$	$fn(s)$
Non-Match	$fp(s)$	$tn(s)$

and the class imbalance ratio. We need to note that, blocking algorithms take records from one or two datasets as input, and classification algorithms take feature vectors being generated from record pairs using comparison functions as input. These feature vectors are generated based on 2-gram Jaccard similarity if not specified.

2.2.2 Measurements

We use the following measures in this thesis to evaluate the quality and efficiency of our approaches. Following the previous work, different quality measures are used for blocking and classification tasks, respectively.

2.2.2.1 Quality Measurements

Blocking quality

We use the measures which are widely used [27] for blocking scheme evaluation in our thesis, which are originally used in Information Retrieval (IR) [127; 105]. Ideally, a good blocking scheme should yield blocks that minimize the number of record pairs to be compared, while preserving true matches at a required level. Given a pair of records that are placed into the same block, we call it a *true positive* if it refers to a match; otherwise, it is a *false positive*. Similarly, a pair of records is called a *false negative* if it refers to a match but the records are placed into two different blocks. For convenience, we use $tp(s)$, $tn(s)$, $fp(s)$ and $fn(s)$ to denote the numbers of true positives, true negatives, false positives and false negatives in blocks w.r.t. a blocking scheme s, respectively. These measures are defined in Table 2.4 using a two-by-two contingency matrix. Based on the notions of these, we adopt the following measures to evaluate blocking quality:

- **Reduction Ratio (RR)** of a blocking scheme s is one minus the total number of record pairs generated by a blocking model in terms of s divided by the total number of record pairs without blocks, which measures the reduction of compared record pairs. That is, RR is calculated as:

$$RR = 1.0 - \frac{tp(s) + fp(s)}{tp(s) + fp(s) + tn(s) + fn(s)} \tag{2.1}$$

- **Pairs Completeness (PC)** of a blocking scheme s is the number of true positives $tp(s)$ divided by the total number of true matches, i.e. $tp(s) + fn(s)$, which measures the accuracy, i.e. rate of matches remained in blocks, is calculated as:

$$PC = \frac{tp(s)}{tp(s) + fn(s)} \tag{2.2}$$

Table 2.5: **The notions tp, tn, fp, fn defined in classification.**

True Label	Predicted Class	
	Class = Positive	Class = Negative
Match	tp	fn
Non-Match	fp	tn

- **Pairs Quality (PQ)** of a blocking scheme s is the number of true positives $tp(s)$ divided by the total number of record pairs that are placed into the same blocks, i.e. $tp(s) + fp(s)$, which measures the efficiency, i.e. rate of true positives in blocks, is calculated as:

$$PQ = \frac{tp(s)}{tp(s) + fp(s)} \tag{2.3}$$

- **F-measure (FM)** Both pairs completeness and pairs quality are essential to evaluate a blocking approach, but the results may be conflict. That is to say, we may notice that the value of PC will decrease when the value of PQ increases in the experiments, thus we need to have a trade-off on PC and PQ. In this scenario, the definition of F-measure was proposed, as the harmonic mean of PC and PQ. Given the value of PC and PQ, the F-measure is calculated as:

$$F - measure = \frac{2 * PC * PQ}{PC + PQ} \tag{2.4}$$

Classification quality

The measures we use to evaluate classification models are defined in Table 2.5 using a two-by-two contingency matrix. The *true positives* are the pairs which are classified as positives and are true matches. The *false positives* are the pairs which are classified as positives but are true non-matches. Similarly, the *true negatives* are the pairs which are classified as negatives and are true non-matches, and the *false negatives* are the pairs which are classified as negatives but are true matches. We can have the following measures in terms of the above definitions.

- **Accuracy:** The fraction of the record pairs in the datasets that are correctly classified by the classification model, which is calculated as:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.5}$$

- **Precision:** The fraction of the number of record pairs classified as matches by the classification model that are true matches, which is calculated as:

$$Precision = \frac{tp}{tp + fp} \tag{2.6}$$

- **Recall:** Recall is the fraction of the number of true matches in the datasets that are correctly classified as matches by the classification model, which is calculated as:

$$Recall = \frac{tp}{tp + fn} \tag{2.7}$$

- **F-measure (FM):** F-measure, which is also called F-score, is equal to the harmonic mean of Precision and Recall. Given the value of precision and recall, the F-measure is calculated as:

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.8}$$

2.2.2.2 Efficiency Measurements

We use the following measures to evaluate the efficiency of entity resolution approaches.

- **Run time:** The computation efficiency of an algorithm can be estimated by the total time required for the ER process.
- **Memory size:** The size of memory required during the entity resolution process.
- **Label cost:** The number of labels used in training a model during the ER process.

Table 2.6: **Notations in this Thesis.**

Notation	Definition
R, $\|R\|$	Dataset and Size of the dataset
a_k, A	Attribute and set of attributes
$r_i.a_k$	Value of attribute a_k w.r.t. record r_i
h	Blocking function
$\langle a_k, h \rangle$	Blocking predicate
P	Set of blocking predicate
T, $\|T\|$	Training set and size of the training set
s, S	Blocking scheme and set of schemes
b_s, B_s	Block and set of blocks generated by scheme s
str	String (a value type)
$dist(str_1, str_2)$	Distance of two strings
$sim(str_1, str_2)$	Similarity of two strings
x_i, X	Sample and set of samples
w_i	Weighted value for a sample x_i
$f(x_i)/g(x_i)$	A function takes a sample x_i as input
$\hat{y}_i$	Predicted label of x_i
y_i, Y	Label, Set of labels corresponding to x_i and X
$v \in \{0, 1\}$	Binary value
$\mathbf{v}$	Binary vector
$G(x_i)$	Label generator for sample x_i
$D(x_i, y_i)$	Discriminator for labeled sample (x_i, y_i)
$p(X)$	Distribution of sample set X
ℓ	Differentialable loss function
l	Loss of a function
ζ	Human oracle
$Budget(\zeta)$	Label budget
$\epsilon \in [0, 1]$	error rate
$\Omega(f)$	Penalty for the complexity of function f
σ	Output vector from a softmax function
Λ	A learning-based model
α	Balancing parameter
$\beta(s, X)$	Balance rate of scheme s in X
Γ	Regularizer for sample distribution
γ	Batch sample size
Δ	Batch sample set
$tp(s)$ / $fp(s)$	True / False positives in blocks B_s
$tn(s)$ / $fn(s)$	True / False negatives in blocks B_s
PC	Pair Completeness
PQ	Pair Quality
RR	Reduction Ratio
FM	F-Measure

Chapter 3

Background and Related Work

This chapter provides the background and related work on entity resolution, especially on scheme-based blocking approaches and machine learning-based classification models. This chapter also gives a brief introduction to skyline queries, active learning techniques and machine learning models which are closely related to the research objectives.

3.1 Entity Resolution

Entity resolution aims to identify different records which refer to the same real-world entity. It plays a more and more important role in many application areas such as linking census records [96], public health [74], web search [75], comparison shopping [151], counter-terrorism [124], and so on in recent years. For example, health researchers are interested in aggregating health datasets from different organizations for quality health analysis such as epidemiological studies and adverse drug reaction investigation. The health-related entity resolution can also be used to develop health strategies in an efficient and effective way, compared with the traditional survey method which is a time-consuming process [33]. Many business companies such as price comparing agents and third-party on-line shopping companies take advantage of entity resolution techniques for matching products from different websites in order to offer a fair price for their products, such as eBay, Amazon. Entity resolution can also be applied to web search areas, for example, identifying documents that belong to the same subject or are written by the same author [114].

In this section, we introduce the procedure of entity resolution in two categories: (1) traditional entity resolution process [26] and (2) the entity resolution process using deep learning-based models [44; 113].

3.1.1 Traditional Entity Resolution

Traditional entity resolution process is composed of four steps: blocking, comparison, classification and clustering, with different techniques being applied in each step. Figure 3.1 outlines these steps, together with the input and output for each step in entity resolution. Date cleaning is a pre-processing step to clean the datasets. For example, if two or more datasets are used for entity resolution, they may have different data structures and are not easy to be used directly, e.g. different attribute names. In addition to this, records to be resolved may contain missing

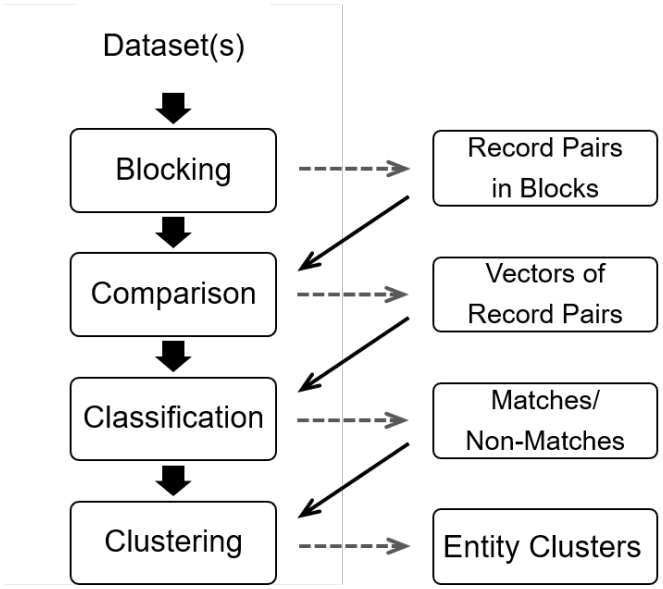

Figure 3.1: The four general steps in entity resolution process with the input and output for each step.

values and noisy data [128] and need to be cleaned. Some techniques of data cleaning for entity resolution are discussed in [32; 26].

Generally speaking, *blocking* is used to reduce the comparison time of record pairs by grouping those potentially matched pairs into the same block. *Comparison* is used to calculate the similarity of a record pair using certain similarity functions, e.g. the similarity of strings, q-gram, TF-IDF, geographical distance and so on. Furthermore, *classification* is necessary to identify if a record pair is a match or not, which can be threshold-based, rule-based, probabilistic-based or machine learning-based techniques [26]. Once all the matches are found, they are grouped into different clusters, all the records within one cluster refer to the same real world entity. *Clustering* can be solved by using the clustering techniques.

3.1.1.1 Blocking

Given two datasets D_1 and D_2, the total comparison time in entity resolution tasks without blocking is $|D_1| * |D_2|$, assuming there is no duplicate in each dataset (i.e., no more than two records referring to the same entity in one dataset). Therefore, with the size of dataset's growing, the comparison time increases quadratically. With the application of blocking techniques, say, if all the records are grouped into k blocks, and the largest block contains $|B_1|$ and $|B_2|$ records from each dataset, respectively, then the total comparison time in the comparison step is no larger than $k * |B_1| * |B_2|$. Similarly, for a single dataset D, the comparison time is no larger than $\frac{k*|B|^2}{2}$, where B is the largest block containing $|B|$ records. Consequently, blocking is an essential step in entity resolution to improve the computational efficiency.

In blocking, attribute values are often used to group records into different blocks [26], which is a straight forward way for the user to understand how the blocks are generated. If

values from more than one attribute are used, they can be in the disjunctive normal form, which refers to a disjunction of conjunctions. In real-world datasets, records may contain mistakes such as typographical errors, and using attribute values for blocking becomes less accurate. Normally, when it is hard to decide whether a similar record pair is a match or not, we would prefer putting them into the same block. This is because, even though grouping records with similar values may lead to placing some true non-matched records into the same block, these records may still be teased out in the further comparison step. On the other hand, if we miss out a true matched pair, it would not be back any more. Hence to avoid this, we use blocking functions considering the similarity of records instead of comparing their exact values. For example, phonetic functions such as soundex [117] can help to alleviate typographical errors, and these functions can be used as blocking functions [27]. In addition to grouping records into disjoint blocks, some approaches adopt a multi-block strategy such that a record can be grouped into multiple blocks [26].

Blocking is regarded as an efficient way to reduce the computational time of the entity resolution process. A number of blocking techniques for entity resolution have been proposed in previous years which have been compared in several surveys [7; 27]. Here we summarise three widely used types as follows.

- **Scheme-based blocking:** Scheme-based blocking approaches are the traditional approaches where each record is inserted into one block only, which is also called standard blocking. In scheme-based blocking, all records that contain the same value referring to the blocking scheme (e.g. same authors in a citation dataset) will be inserted into the same block, and only the records within the same block will be compared in the comparison step [47].

- **Scheme-agnostic blocking:** There are also some scheme-agnostic blocking approaches: (1) Token based blocking, which is also called Q-gram based blocking, aims to group all records containing the same token (e.g., string of length q) into the same block [7; 119]. Rather than generating blocks according to blocking schemes, these approaches can generate blocks w.r.t. tokens called sorting key values (SKVs), which are generally substrings of records [65; 89]. (2) Clustering based blocking [120] aims to group a number of records into the same block by pre-defined criteria using some computationally cheap clustering techniques in an unsupervised manner. These criteria can be similarity thresholds [107], a number of k nearest neighbors [28], a fixed blocking size [50] and so on .

- **Meta blocking:** Different blocking approaches may generate various blocks. Generally speaking, if the size of most of the blocks is large, the number of comparison is often large; if the size of blocks is small, the completeness and quality of blocks can potentially be low. Hence several methods have been proposed to improve the performance of blocking by controlling the block size using meta blocking techniques [122; 121].

In the following, we will review some existing work of these types, respectively.

Scheme-based blocking

Scheme based blocking for entity resolution was first mentioned by Fellegi and Sunter [47]. They are the first to use an ⟨ attribute, blocking-function ⟩ pair as a blocking predicate to define blocks. For example, the soundex code of both names *Gail* and *Gayle* is "G4", thus records that contain either of the names are placed into the same block. However, a blocking scheme, which is defined as a disjunction of conjunctions of block predicates, has to be chosen by domain experts. Michelson has proposed a machine learning-based blocking scheme learning algorithm called *Blocking Scheme Learner (BSL)* [110]. This is the first algorithm to learn blocking schemes instead of generating them by a domain expert. It uses the Sequential Covering Algorithm (SCA) to learn schemes aiming to achieve both high pairs completeness and high reduction ratio. Later on, a number of scheme learning approaches have been proposed, which generally fall into two categories: (1) supervised blocking scheme learning approaches [108; 17], and (2) unsupervised blocking scheme learning approaches [84; 85; 86].

Suffering from the training samples with imbalanced classes, where non-matches are much more than matches, Bilenko et al. [11] proposed two blocking scheme learning algorithms called *ApproxRBSetCover* and *ApproxDNF* to learn disjunctive blocking schemes and DNF (i.e., Disjunctive Normal Form) blocking schemes, respectively. A training set in this approach contains two kinds of pairwise supervision training samples: positive record pairs and negative record pairs. In the ApproxRBSetCover algorithm, given a set of candidate predicates, this algorithm selects those predicates which can cover more than a given number of positive pairs and can not ignore a given number of negative pairs. In the ApproxDNF algorithm, each conjunction of predicates must match the above two criteria with one more constraint: the length of the conjunction must be less than a specific length.

Additionally, one challenge in entity resolution is that obtaining the labels for all possible record pairs is very expensive. To deal with this challenge, Cao et al. [17] used both labeled and unlabeled samples in their approach to improve the label efficiency compared with other works. Their algorithm can learn a blocking scheme using conjunctions of blocking predicates to satisfy both minimum true-match coverage and minimum precision criteria.

Later on, in 2013, Kejriwal et al. [84] proposed an unsupervised algorithm for learning blocking schemes. In this paper, a weak training set was considered. However, the labels of record pairs (positive or negative) in a training set are not generated manually, but by calculating the similarity of record pairs in terms of their TFIDF statistics. A training set also contains two parts: positive labeled pairs and negative labeled pairs. An algorithm called *FisherScore* [11] is proposed to calculate the fisher score for each given attribute. In the main algorithm called *FisherDisjunctive*, two criteria are used the same as in *ApproxRBSetCover*, and predicates that match the criteria are sorted by their fisher scores in descending order. The first predicate is selected as part of the blocking scheme, and if the later predicate can cover at least one new positive pair in the training set, it will be selected in a disjunctive form. After all the positive pairs are covered by the disjunction of predicates, the blocking scheme is learned.

Scheme-agnostic blocking

Hernandez et al. proposed *sorted neighborhood blocking* as a scheme-agnostic blocking approach [65]. This approach uses a 'sorting key' to sort records according to the sorting key value (SKV), over which a sliding window of fixed size is then used and candidate record pairs are generated from records that are within the current window. The sorted neighborhood

blocking approach is not suitable if the first character of a SKV contains errors. *Q-gram based blocking* has been proposed to overcome this drawback by generating variations of each attribute value using q-grams (sub-strings of length q), and inserting record identifiers into more than one block [89; 97; 144]. *Suffix array based blocking* has been also proposed to overcome the issue of errors and variations at the first character of a SKV by generating suffix substrings of blocking predicate values, which are called suffixes [1]. The suffixes of a string are sub-strings with one or more characters at the beginning removed. Additionally, *string-Map based blocking* aims to map SKVs to objects in a multi-dimensional Euclidean space while preserving the similarities (distances) between blocking predicate values [77].

Some existing work has formulated blocking as an unsupervised clustering problem, where similar records are placed into the same cluster. The *Canopy clustering* technique for blocking is based on the idea of using a computationally effective and efficient clustering technique to create high-dimensional overlapping clusters, from which candidate record pairs can be generated [35; 153]. The state-of-the-art blocking technique in this line was proposed by Fisher et al. [50].

Meta blocking

Whang et al. [152] proposed an iterative blocking framework where blocks are iteratively processed until no block contains any more matching records, i.e., if one record appears in a block, all its matched records will be presented in the same block. In this framework, two algorithms called *Lego* and *Duplo* have been proposed, while the second one was used for large scale datasets which have to be stored in disk instead of memory.

Papadakis et al. proposed an unsupervised meta-blocking framework [122] which aimed to extract the most similar pairs of records from blocks. This framework first generates a blocking graph based on existing blocks. In such a graph, each node represents a record, and a weighted edge between two nodes refers to the similarity of the corresponding records. Then, four algorithms have been proposed to prune unnecessary edges and nodes based on edge weights. *Weight Edge Pruning* and *Cardinality Edge Pruning* are two algorithms to prune edges in order to change record pairs from positives (predicted as matches) to negatives (predicted as non-matches). The former algorithm uses the minimum edge weight as a threshold, while the latter algorithm reserves top-k highest edge weights. *Weight Node Pruning* and *Cardinality Node Pruning* are two local pruning algorithms to discard edges of a specific node based on the edge weights between this node and its neighbors. The former algorithm uses the local minimum edge weight as a threshold, while the latter algorithm reserves top-k nearest nodes for this specific node, i.e., k nodes with highest edge weights.

Since the unsupervised meta-blocking framework proposed in [122] can not provide a quality guarantee, which can be either conservative (i.e., fail to discard unnecessary record pairs) or aggressive (i.e., discard necessary record pairs), Papadakis et al. proposed another supervised meta-blocking framework [121]. This framework aims to learn an optimal classifier to decide the best k-value for cardinality pruning and the edge weight threshold for weight pruning.

3.1.1.2 Comparison

Comparison is an essential step in the entity resolution process to compare the similarity for each record pair. A number of comparison techniques have been proposed based on measuring

different kinds of distances of two records whose values are regarded as two strings. Some of the widely applied comparison techniques are listed as below, where str_1, str_2 refers two strings corresponding to two records r_1 and r_2.

Exact Comparison: A simplest method to compare a record pair $\langle r_1, r_2 \rangle$ is to consider two records as two strings $\langle str_1, str_2 \rangle$ and check whether they are the same. The comparison result of a record pair is considered as 1 if two strings are identical, otherwise it is 0. The exact comparison function is:

$$sim_{exact}(str_1, str_2) = \begin{cases} 1 & \text{if } str_1 = str_2 \\ 0 & \text{if } str_1 \neq str_2 \end{cases} \tag{3.1}$$

Truncate Comparison: If the string value of a record is too long or contains some errors in some substrings (this is because a string may contain several attribute values corresponding to a record, and some values may contain errors), we can truncate the string and use only the beginning or ending characters. If the first i characters of a string with n characters are denoted with $str[1:i]$ and the last j characters are denoted with $str[j:n]$, then a truncate function can be described as:

$$sim_{trun\_begin(i)}(str_1, str_2) = \begin{cases} 1 & \text{if } str_1[1:i] = str_2[1:i] \\ 0 & \text{if } str_1[1:i] \neq str_2[1:i] \end{cases} \tag{3.2}$$

$$sim_{trun\_end(j)}(str_1, str_2) = \begin{cases} 1 & \text{if } str_1[j:n] = str_2[j:n] \\ 0 & \text{if } str_1[j:n] \neq str_2[j:n] \end{cases} \tag{3.3}$$

These functions, i.e., Functions 3.1, 3.2 and 3.3, are proposed based on the exact record values, thus they are also called binary comparison techniques. However, in real-world datasets for entity resolution, records are not "clean" to generate reliable feature vectors based on their exact values. To deal with such real-world datasets, we apply approximate comparison techniques which present how similar two records are by calculating the distance between them. Such approximate comparison techniques using similarity functions can return a normalized similarity value $sim \in [0, 1]$, where 0 means that two records are totally different and 1 means that two records are exactly the same.

Levenshtein Edit Distance String Comparison: The basic edit distance is also called Levenshtein edit distance [78; 115]. The Levenshtein edit distance of a record pair is defined as the minimum number of edit operations one string str_1 need to convert into the other string str_2. Here, an edit operation is an single character operation including deletion, insertion and substitution. The record pair similarity based on Levenshtein edit distance can be presented as:

$$sim_{Levenshtein}(str_1, str_2) = 1 - \frac{dist_{Levenshtein}[str_1, str_2]}{max(|str_1|, |str_2|)} \tag{3.4}$$

For the first i characters in str_1 and the first j characters in str_2:
If $str_1[i] = str_2[j]$, then

$$dist_{Levenshtein}[i, j] = dist_{Levenshtein}[i-1, j-1]$$

If $str_1[i] \neq str_2[j]$, then

$$dist_{Levenshtein}[i,j] = minimum \begin{cases} dist_{Levenshtein}[i-1,j]+1 & \text{a deletion,} \\ dist_{Levenshtein}[i,j-1]+1 & \text{an insertion, or} \\ dist_{Levenshtein}[i-1,j-1]+1 & \text{a substitution.} \end{cases} \quad (3.5)$$

Example 3.1.1. *The Levenshtein edit distance between kitten and sitting is 3: (1) starting with kitten, we first replace k by s, obtaining sitten; (2) then the 5-th character e is replaced by i, obtaining sittin; (3) finally, an insertion operation of g is used at the end of the string, obtaining sitting.*

Smith-Waterman Edit Distance String Comparison: Another edit distance based approximate comparison technique is called Smith-Waterman edit distance [115], which calculates the record pair similarity as:

$$sim_{Smith\_Waterman}(str_1, str_2) = \frac{bs_{Smith\_Waterman}}{div_{Smith\_Waterman} \times ms_m} \quad (3.6)$$

where ms_m is one of five scores representing five basic character operations, including exact match (score of 5), approximate match (score of 2), non-match (score of -5), missing value start (score of -5), and missing value continuation (score of -1). $div_{Smith\_Waterman}$ can be decided by one of the three values: $min(|str_1|, |str_2|)$, $max(|str_1|, |str_2|)$, or $\frac{|str_1|+|str_2|}{2}$. $bs_{Smith\_Waterman}$ uses the same matrix for calculating deletion, insertion or substitution as in Equation 3.5, but it uses the maximal value instead of the minimum value among them.

These functions, i.e., Function 3.4 and 3.6, are distance-based, where we calculate the distance between two records by regarding them as strings. The similarity is 1 minus the distance between them. Three main properties of distance functions should be noticed that: (1) all distance values must be non-negative; (2) the distance between two strings i and j is symmetric, i.e., $dist(i,j) = dist(j,i)$; (3) the distance between i and j must no larger than the combined distance between them via a third object k, i.e., $dist(i,j) \leq dist(i,k) + dist(k,j)$, which is also called the triangular inequality property.

Q-gram based String Comparison: We discuss how q-gram based techniques are used for record comparison. We split a record pair to be compared into two sets of sub-strings, one set for one record, and each sub-string contains q characters, for example, 3-gram sub-strings of "elizabeth" are {"eli", "liz", "iza", "zab", "abe", "bet", and "eth"}. The similarity of two records can be calculated by the following comparison functions:

$$sim_{overlap}(str_1, str_2) = \frac{|str_1 \cap str_2|}{min(|str_1|, |str_2|)} \quad (3.7)$$

$$sim_{Jaccard}(str_1, str_2) = \frac{|str_1 \cap str_2|}{|str_1 \cup str_2|} \quad (3.8)$$

$$sim_{dice}(str_1, str_2) = \frac{2 \times |str_1 \cap str_2|}{|str_1| + |str_2|} \quad (3.9)$$

These three functions calculate the overlap similarity, Jaccard similarity and Dice similarity [26] of a record pair.

Monge-Elkan String Comparison: This comparison technique has been proposed specifically to calculate the similarity of strings of words such as business names, addresses and personal names that are not standardized and segmented [111]. This comparison technique works as follows: first two records are split into two sets A and B of word tokens, then each token in one set is compared with all the tokens in the other set in terms of a similarity function (where it is called the secondary similarity function to distinguish the one for Monge-Elkan similarity). Finally, the maximal similarity value of each token in set A corresponding to all the tokens in set B is selected to calculate to Monge-Elkan similarity.

SoftTFIDF String Comparison: Since the TF-IDF (Term Frequency and Inverse Document Frequency) technique has been widely used for document analysis, Cohen et al. introduced it as a comparison technique in entity resolution [34]. The concept of term frequency is used to identify relevant terms which have a high frequency, and the inverse document frequency is used to identify terms that can distinguish documents with low frequency. In the record comparison, a word token is regarded as a term.

3.1.1.3 Classification

A number of classification approaches for entity resolution have been proposed in the literature [25; 66; 26]. Traditional classification techniques can be grouped into the following categories:

- **Threshold-based techniques:** With the vectors of record pairs obtained as the output of the comparison step, we can set a threshold to classify all the record pairs into two categories: match and non-match. The record pairs whose vector values are higher than the threshold are regarded as matches. In some cases, an upper threshold as well as a lower threshold are used, and record pairs whose vector values are above the upper threshold, between two thresholds or below the lower threshold are considered as matches, potential matches or non-matches, respectively [25].

- **Probability techniques:** The probability techniques are similar to the threshold-based techniques in that we also need to set one or two thresholds. However, the difference is that we need to calculate the probability of being considered as a match or non-match for each record pair in terms of its vector values [51; 66].

- **Cost-based techniques:** There are two kinds of errors when we consider if a record pair is a match or non-match, i.e., a record pair referring to the same real-world entity is classified as a non-match, or a record pair referring to different real-world entities is classified as a match. The probability techniques consider these two types of errors equally important. However, we can still weight different types of probabilities by adding a cost for each probability [145]. For example, we can add a higher cost to the probability of classifying a true non-match pair as a match, or we can add a lower cost to the probability of classifying a true match pair as a potentially match.

- **Rule-based techniques:** The above techniques consider each vector as a whole when they classify the corresponding record pair as a match or non-match. However, with the application of rule-based techniques [150; 46], we can consider each value in a vector separately by adding rules with different thresholds for each vector value we need. For example, we have vectors of n values $(sim_1, sim_2, ..., sim_n)$, and we define the classification rules as $(sim_1 \geq 0.8 \wedge sim_2 \leq 0.2 \wedge sim_4 \geq 0.4)$ or $\big((sim_1 \geq 0.6 \vee sim_2 \leq 0.4) \wedge (sim_4 \geq 0.4 \vee sim_6 \geq 0.75)\big)$. Such hand-crafted rules require certain thresholds to define similarity or dissimilarity of records [25; 137].

Additionally, **learning-based techniques** are widely used which adopt a machine learning model to classify whether a feature vector of a record pair refers to the same entity. In the literature, there are three main categories: (1) Supervised learning approaches, which train a model with labeled samples so that it can predict unlabeled samples. The most recent work is *Magellan* [92], which considered learning models including *Decision Tree*, *Random Forest* and *Support Vector Machine* (SVM). Some work also studied ensemble learning approaches by building a strong learner based on a set of weak learners [52]. A widely used ensemble classifier is *extreme gradient boosting* (XGBoost) [19], which used the sparsity-aware algorithm and the weighted quantile sketch for approximate learning. (2) Unsupervised learning approaches, which do not consider any ground truth labels, but assign labels to samples based on prior knowledge such as record similarity [10; 79]. One approach for entity resolution is called *two-steps* (2S) [24]. It first labeled a number (e.g. 10 percents of a dataset) of the most similar and dissimilar record pairs, respectively, and then trained an SVM in the second step. A recent work in this line was proposed by Jurek et al. [79], which considered both ensemble learning and automatic self-learning for classification based on training labels which are automatically generated w.r.t. different similarity measure schemes. A graph-based unsupervised approach for entity resolution was proposed by Zhang et al. [158] which has two components: Iterative Term-Entity Ranking (ITER) and CliqueRank for record graph construction. (3) Semi-supervised learning approaches, which sit between supervised and unsupervised learning in that they take a limited number of real labeled samples and sufficient unlabeled samples for training. The state-of-the-art semi-supervised learning approach for entity resolution is an ensemble learning-based approach using Adaboost [129] for label prediction based on seed samples that have real labels [87].

3.1.1.4 Clustering

Once we obtain the predicated labels for all the record pairs, i.e., matches and non-matches, we can resolve entities with corresponding records, where each matched pair refers to exact one real-world entity. However, what is the efficient way to find the entity if we have three records $\langle r_1, r_2, r_3 \rangle$, where $\langle r_1, r_2 \rangle$ and $\langle r_1, r_3 \rangle$ are matches, but $\langle r_2, r_3 \rangle$ is a non-match? What about more than three records referring to the same entity? We need clustering techniques. Many clustering techniques are developed by researchers from statistics, data mining and machine learning domains, and they use different heuristics to guide the clustering process [64]. Clustering techniques in entity resolution group records into different entity clusters, and each cluster contains records corresponding to the same entity [148; 147]. Clusters may be small, or

even contain one record, which means only one record in the dataset(s) referring to this entity [147].

Lokhande et al. [104] proposed a correlation-clustering based approach. In this approach, the clustering problem is first treated as a Minimum Weighted Set Packing (MWSP) problem for optimization. Furthermore, the MWSP problem is tackled using Column Generation (CG), which targets at solving an Integer Linear Programming (ILP) of the MWSP by constructing a small sufficient subset of samples. Additionally, the Flexible DOIs (F-DOIs) has been proposed based on the accelerated CG using Dual Optimal Inequalities (DOIs) technique, which helps to reduce the search space of the LP problem.

3.1.2 Entity Resolution with Deep Learning

In recent years, motivated by the success of deep learning techniques in computer vision [60; 59], natural language processing [39] and so on, several attempts have been made to design deep learning solutions for entity resolution tasks [30]. In such approaches, traditional comparison and classification steps are merged into one: the representations of records are first learned, then they are compared and aggregated in a sequence, and finally a model is used to predict the labels corresponding to the record pairs, i.e., matches or non-matches [116].

Representation learning and comparison. Ebraheem et al. proposed DeepER, which used bi-directional Recurrent Neural Networks (RNNs) with Long Short Term Memory (LSTM) units to learn a distributed representation for each record [44]. Mudgal et al. studied how to use deep learning techniques developed in natural language processing to handle the problems of attribute embedding, attribute summarization and attribute comparison [113]. A recent work proposed by Nie et al. [116] used an align-compare-aggregate framework for a token level sequence-to-sequence entity resolution which aimed to solve the heterogeneous and dirty data problems, specifically, it learned the representations of tokens, captured the semantic relevance between tokens, and aggregated matching evidence for accurate entity resolution decisions in an end-to-end manner. In deep learning approaches, the comparison step of traditional entity resolution is replaced by a comparison layer. However, there is no specific measure for record comparison. Fu et al. [56] proposed an end-to-end multi-perspective entity matching model, which can adaptively select optimal similarity measures for heterogeneous attributes by jointly learning and selecting similarity measures in an end-to-end way.

Deep learning under limited labels. Several approaches have been proposed to deal with the problem of insufficient labels in entity resolution. Taking the advantages of transfer learning techniques, Zhao and He [159] proposed Auto-EM, which leverages pre-trained entity resolution models from large scale, production knowledge bases. In this model, for each entity type in the knowledge base, such as people and location, the synonymous names of known entities are used for pre-training, thus models for each type are trained using a hierarchical neural network architecture. Additionally, with little or no training data, i.e., either fine-tuning or using the pre-trained entity resolution directly, a target entity resolution task can be solved. Kasai et al. also proposed an entity resolution solution with both trasfer learning and active learning [81]. However, the well-known limitations of transfer learning are that (1) it needs a pre-trained model before applying to a target task, and (2) a prior assumption on the correlation between the source and target tasks is also required, which restrict its practical applicability for

entity resolution problems in real-world applications. Some more approaches are reviewed by Christophides et al. [30]. The state-of-the-art unsupervised approach is called *ZeorER*, which is proposed by Wu et al. based on the insight that the feature vectors of matches are different from those of non-matches [154]. This approach is built based on a generative model w.r.t. Gaussian Mixture Models to learn the distribution of matches and non-matches. Additionally, since samples used to train the model are not labeled manually, but based on feature vectors, to avoid the extreme cases for feature overfitting, e.g. dis-similar vectors may be considered as matches by the model, an adaptive regularization technique is proposed. Finally, the transitivity property is used in the generative model for performance improvement.

Deep learning for unstructured data. While some of the approaches are limited to resolve entities under specific data structures, i.e., schema-specific, taking the advantage of pre-trained language models such as BERT [39], Teong et al. [142] proposed a scheme-agnostic model. This model is first pre-trained as a language model, i.e., BERT and then fine-tuned by the labeled entity resolution dataset. Li et al. [102] proposed an entity matching system based on pre-trained transformer-based language models called DITTO by fine-tuning and casting EM as a sequence-pair classification problem to leverage such models.

Deep graph-based entity resolution. Li et al. [99] is the first to deal entity resolution tasks in the token-centric manner, i.e., to present and compare records based on their token values, using an entity record graph. It helps to overcome some shortages of attribute-centric approaches. These shortages can be the semantic sparsity and information dilution problem of attribute representations, the inflexible comparison problem from hard attribute alignment and the difficulty in handling heterogeneous attributes. The proposed model called *GraphER* is composed of four layers. The ER-GCN layer is built based on Graph Convolutional Networks (GCNs) to capture both the semantic and structural information of attribute values, and embed them into token representations. The comparison layer yields comparison vectors by comparing the representations of record pairs. Then the aggregation layer is used to find the important matching features. Finally the prediction layer is used to predict the final labels for record pairs.

3.2 Skyline Queries

Skyline queries normally refer to finding a set of objects that are useful to a user, where these objects belong to a multi-dimensional space. Typically, there exist trade-offs between these dimensions, so that there is no unique object to be the best [88]. A good number of approaches on skyline queries have been studied in the context of database in the literature [21; 103; 141; 131]. It can be regarded as the queries of value-for-money problem [80], which provides a set of options representing the optimal combinations of the characteristics of a database, for example, the location and price of all the hotels in a city. Existing approaches primarily focused on learning representative skylines, such as top-k RSP (representative skyline points) [103], k-center (i.e., choose k centers and one skyline point for each center) [141] and threshold-based preference [131].

From an algorithmic perspective, a naive algorithm for skyline queries (e.g., nested-loop algorithm) has the time complexity $O(n^2d)$, where n is the number of records and d is the

number of attributes in a given database. Later on, several algorithms have been proposed to improve the efficiency of skyline queries based on different properties which have been previously ignored. In the early days, Borzsony et al. [13] proposed the BNL (block nested-loop) algorithm based on the transitivity of a dominance relation (e.g. if a dominates b and b dominates c, then a dominates c). Then, Chomicki et al. [22; 23] proposed an SFS (sort-filter-skyline) algorithm with the improvements: progressive and optimal comparison times. Sheng and Tao proposed an EM (external memory) model based on an attribute order [13]. Morse et al. proposed the LS-B (lattice skyline) algorithm based on the low cardinality of some attributes (e.g. the rating of a movie is an integer within a small range of [1, 5]) [112]. Papadias et al. proposed the BBS (branch-and-bound skyline) algorithm based on the index of all input records by an R-tree [123].

Existing work on skyline queries aims to efficiently tease out a skyline of queries over a database in which records and attributes are known. In contrast, our study has shifted the focus to learning a skyline of blocking schemes in terms of a given number of selected criteria but the actual values w.r.t. these criteria are not directly available in a database. Particularly, in many real-world applications, only a limited number of labels are allowed to be used for assessing blocking schemes. Thus, how to efficiently and effectively learn a skyline of blocking schemes is a difficult task. In this thesis, different from previous works on blocking schemes and skyline queries, we consider to leverage active learning techniques for finding informative samples and improving the performance of learning, and propose novel algorithms to efficiently learn skylines of blocking schemes .

3.3 Active Learning

In this section, we first briefly introduce the general active learning techniques which are shown to be efficient in reducing the label usage and overcoming the class imbalance problem (i.e., two main issues in entity resolution). Then we introduce some entity resolution approaches that have adopted the active learning techniques and how the state-of-the-art learning-based active learning technique is developed.

Active learning in general. Active learning has been extensively studied in the past [134]. The goal of active learning is to enable a machine learning-based model, where large amounts of training samples are normally required, to achieve better performance with relatively fewer but representative training samples, especially when the labels are expensive and very hard to obtain. These samples may be selected from an unlabeled dataset by posing queries and then asking labels from an oracle [133]. Dasgupta and Hsu have analyzed sampling bias and proposed a clustering-based active learning approach for hierarchical sampling with proved statistical properties [37]. An algorithm called *Cluster-adaptive active learning* was proposed in this active learning approach. This algorithm first randomly selects several samples; then, it selects samples whose probability belongs to a specific range, where such samples are assumed not to be pruned. The probability is calculated by an active learning rule which helps to reduce sampling size. After each cluster is generated by the active learning sampling algorithm, this algorithm finds the observed majority label in each cluster, and assigns this label to all samples in this cluster. This approach is proved to be statistically consistent and have lower label

complexity than supervised learning by selecting biased samples for training. General active learning techniques have been extensively reviewed in [133].

Uncertainty sampling and diversity sampling. Among various active learning techniques, uncertainty sampling is one of the widely used, which was first proposed by Lewis and Gale [98]. Normally, uncertainty sampling approaches select samples by measuring their uncertainty, such as probabilistic confidence [36], fisher information [133], entropy [68] and so on. This technique is usually associated with a probabilistic learning model in order to infer labels with the highest probability [90; 126]. A common issue of uncertainty sampling approaches, although computationally efficient and simple to use, is that they do not consider the diversity of data, for example, data with imbalanced class distribution [45]. Furthermore, most of existing uncertainty sampling techniques have the limitation that a sample can be an uncertain sample to one class but a certain sample to another class [72].

Diversity sampling is also a useful technique in active learning [16; 156], which aims to select representative samples according to the data distribution. In practice, while uncertain samples are often similar to each other [157], diversity sampling requires samples to be dissimilar in certain features. Thus, samples from different groups or classes are more preferred. In this thesis, we adopt the $l_{2,1}$ norm [76] as a measure for diversity sampling.

Active learning for the class imbalanced problem. One of the critical issues in entity resolution is the class imbalanced problem, where there are more true non-matches than true matches for resolving datasets [26; 49]. Ertekin et al. [45] showed that active learning can provide almost the same or even better results in solving the class imbalance problem, in comparison with the oversampling and/or undersampling approaches which also aim to alleviate the class imbalanced problem in entity resolution [18].

In 2013, Ferdowsi et al. proposed an active learning approach to deal with the imbalance class problem [48], which used an unsupervised score called *Mean Score on the Unlabeled set (MSU)* to switch between different candidate *Instance Selection Strategies (ISS)* for classification in imbalanced datasets. The MSU score is obtained based on the mean score of top k% ranked examples from the unlabeled pool where k varies according to the evaluation metric. The authors have a hypothesis that if a classifier gets better, the MSU calculated based on the top k% samples will increase on average. Thus, the score may change during each iteration of the active learning process and the best will be selected. Finally it selects the best ISS in each iteration according to the values of the MSU score.

Learning-based active learning. Despite a large number of studies on developing active learning approaches, it is still difficult for a specific task to determine its best-suited one. Thus, meta-learning algorithms have attracted much attention in recent years, driven by the desire to automate the selection process of active learning approaches. These approaches are also called learning-based active learning approaches, which were proposed to deal with such limitation when pre-defined heuristics become less useful during the model training process [69; 93]. In these approaches, the estimated performance of a current model is used as heuristics instead of pre-defined heuristics for sampling. Two kinds of learning-based active learning approaches have been proposed in the literature: One learns to select active learning strategies for a given dataset [69]; The other builds a machine learning model to rank samples for selection [93].

Hsu and Lin [69] proposed *Active Learning by Learning* (ALBL) which relates active

learning with a multi-armed bandit learner. This approach aims to learn from the performance of a set of active learning strategies (e.g. algorithms) adaptively so as to decide which is the best. Specifically, it actively selects samples iteratively in terms of their voting scores from a set of active learning algorithms by solving the multi-armed bandit (MAB) problem, where each arm is an active learning algorithm. Furthermore, the voting score of a sample in a given iteration is the sum of the probabilities of selecting this sample by each algorithm times the weight vector of each algorithm. Thus, it can be calculated in two steps. Firstly, the weight vector of each algorithm is calculated, which can be analogized as the reward for bandits in MAB. Secondly, the probability of selecting a sample and querying its label for each algorithm is calculated. Chu and Lin extended this work by using LinUCB (Linear Upper-Confidence-Bound) as a measure instead of IW-ACC to calculate the rewards for different active learning strategies. LinUCB is a state-of-the-art technique in balancing the exploration and exploitation, i.e., the estimated reward of using a strategy and the uncertainty of using this strategy. This extension can also transfer the experience on active learning strategies from one dataset to different datasets [31].

The key idea of a recent work called *Learning Active Learning* (LAL) [93] is to train a regressor which can predict the generalization error reduction of each unlabeled sample and greedily select one with highest error reduction for labeling. This regressor can be trained as follows: First, given two training sets differing in only one sample, a pair of classifiers is trained, and the corresponding error reduction value of the sample is obtained. Second, the parameters from different pairs of classifiers and the corresponding error reduction values are collected using the Monte Carlo method [63] to train the regressor.

There are several other approaches named with "learning to sample". For example, Li et al. [100] proposed a generative adversarial network (GAN) based sampling approach which learns to generate synthesized samples by learning likelihood ratios. This approach can also learn to draw samples from an un-normalized distribution via a reference distribution or using Markov Chain Monte Carlo (MCMC). Jamshidi et al. [73] proposed a transfer learning-based approach, which learns the changing of each environment repeatedly for sample selection in configurable software systems. Dovrat et al. [42] proposed an approach to simplify 3D point clouds by matching them to a fixed size of samples via a learned deep network. However, all these approaches do not specifically focus on developing active learning techniques.

In summary, existing work still has the limitations that: (1) When the training samples are not sufficient, they use synthetic data with simple features to train a regressor for entropy prediction, not real data. (2) They only consider the uncertainty of samples and focus on dealing with binary classification problems. Particularly, as reported in [157], uncertain samples are often similar to each other, i.e., the neighbours of an uncertain sample are also of high uncertainty. Additionally, none of the existing work aims to solve entity resolution tasks using learning-based active learning techniques.

3.4 Ensembling Techniques for Classification

Bagging [15] is a parallel ensembling technique which refers to bootstrap aggregation based on bootstrap sampling technique:

$$f(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(x)$$

Where M is the number of classifiers $f_m(x)$. The training data for each classifier is a subset of the whole training set using bootstrap sampling (drawn with replacement). The final output of a bagging model is a voting score for classification or the averaging score for regression. Random forest [67; 140] can be regarded as an extension of bagging, which uses not only a random subset of training data, but also a random subset of features for each classifier. Thus it can reduce both variance and bias w.r.t. the bagging technique and the CART (Classification And Regression Tree) [14] structure.

Boosting is a sequential ensembling technique which refers to a family of algorithms that aim to build a strong classifier w.r.t. a set of weak classifiers sequentially. In boosting, weights are assigned to samples or trees iteratively during the process of training weak classifiers by re-weighting. The final output of a boosting model can be a weighted majority vote for classification or a weighted sum values of each individual classifier for regression. A number of *boosting* techniques have been proposed which use a set of weak learners (e.g. decision tree and SVM) to create a single strong learner [83]. Freund developed the first boosting algorithm [52]. Later on, the first adaptive boosting approach, called *AdaBoost*, was proposed [54], in which the parameters of a model can be self-adjusted based on the actual performance in each iteration, including weights for samples and weights for additive learners. Compared with *AdaBoost*, which favors on dealing with classification tasks, *Gradient Boosting* [55] approaches were proposed to solve both classification and regression problems by reducing the loss of a model in a gradient descent manner.

For example, AdaBoost with additive model aims to minimize the exponential loss:

$$f(x) = \sum_{m=1}^{M} a_m f_m(x)$$

Specifically, in boosting algorithms such as Adaboost and Gradient boosting, a general way to minimize the loss is to train each additive function $f_i(x)$ which aims to reduce the residuals $y - \sum_{m=1}^{i-1} f_m(x)$ of the predictions from previous functions as its objective.

The state-of-the-art and widely used gradient boosting approach is *XGBoost* [19]. With the use of the sparsity-aware algorithm and the weighted quantile sketch for approximate learning, *XGBoost* can deliver results more accurately and efficiently than previous work. There are also some other boosting models in the literature: Light GBM [82] performs well when the dataset is extremely large, which uses a leaf-wise tree growth strategy instead of level-wise tree growth in other models; CatBoost [41] targets at handling high dimensional data, such as large numbers of categorical variables that need to be one-hot encoded.

3.5 Generative Adversarial Networks

Generative adversarial network (GAN) was proposed by Goodfellow et al. [60]. The key idea of GAN is that two networks, a *generator* and a *discriminator*, play a minimax game so that they converge gradually to an optimal solution. The generator aims to generate fake samples to "fool" the discriminator by simulating the distribution of real samples, while the discriminator targets to distinguish fake samples (generated by the generator) from real samples. Due to the success of GAN in generating realistic images, a large number of studies have extended GAN to dealing with various tasks such as sample classification with semi-supervised learning [138; 130], labeled sample generation [109] and label generation [38]. Various techniques have also been proposed to improve GAN's performance by alleviating the mode collapse and convergence problems [130; 5].

The objective function of the original GAN plays a minimax game as:

$$\min_G \max_D \quad \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x\sim p_z(z)}[1-\log D(G(z))]$$

where $p_{data}(x)$ refers to the distribution of real samples, and $p_z(z)$ refers to the distribution of generated samples.

Later on, a fixed objective function using "-log D trick" was proposed [59], where the loss of the discriminator is not relied on the generator:

$$\min_G \max_D \quad \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x\sim p_z(z)}[-\log D(z)]$$

There are also some other techniques and tricks used to improve GANs performance, such as feature mapping [130]. Although GAN-based techniques are exploding, they cannot be directly used in solving entity resolution tasks for three reasons: (1) entity resolution datasets are often highly imbalanced, which aggravates the need of sufficient labeled training data, and may cause the mode collapse problem during the training process; (2) Most of the GAN-based approaches, including the ones designed for semi-supervised learning [138], have not considered the case of training with an extremely limited number of real labeled samples; (3) Traditionally, the generator in GANs is designed to generate new samples; however, for entity resolution tasks, classifying all unlabeled samples is the ultimate goal. In this thesis, we build ERGAN which will be presented in Chapter 7 to fill in this gap.

Chapter 4

Active Blocking Scheme Learning for Entity Resolution

4.1 Introduction

Blocking is an important process in entity resolution which helps to improve the time efficiency by grouping potentially matched records into the same block at the beginning of the entity resolution tasks. In this chapter, we study the problem of how to learn a blocking scheme efficiently under a limited number of labeled samples with quality guarantees.

In recent years, both supervised and unsupervised approaches have been proposed for blocking scheme learning, such as *Blocking Scheme Learner (BSL)* which targeted at automatically learning effective blocking schemes [108] and *Fisher* which used similarity based labels for record pairs to build a training set w.r.t. the TF-IDF measure, and a blocking scheme can then be learned from a training set [84]. However, these existing approaches on blocking scheme learning still have some limitations:

- It is expensive to obtain ground-truth labels in real-life applications. Particularly, match and non-match labels in entity resolution are often highly imbalanced [146], which is called the *class imbalance problem* and aggravates the cost of labels in blocking scheme learning. For example, given a dataset with two tables, each table containing 1,000 non-duplicate records, the total number of record pairs will be 1,000,000, but the number of true matches is no more than 1,000. The class imbalance ratio of this dataset is thus at most 1:1,000. This indicates that the probability of randomly selecting a matched pair from this dataset is 0.1%. Existing supervised learning approaches use random sampling to generate blocking schemes, which can only guarantee the blocking quality when sufficient training samples are available [108].

- Blocking quality is hard to be guaranteed in unsupervised approaches. These approaches obtain the labels of record pairs based on the assumption that the more similar two records are, the more likely they can be a match. However, this assumption does not always hold [147]. As a result, the labels may not be reliable and no blocking quality can be promised.

- The search space for all possible blocking schemes is huge. As will be discussed in

Section 4.4, if a blocking scheme is composed of at most n different blocking predicates, the number of all possible blocking schemes can be $O(2^{\binom{n}{\lfloor n/2 \rfloor}})$ asymptotically.

A question arising is: Can we learn a blocking scheme with blocking quality guaranteed and the cost of labels reduced? To answer this question, we propose an active blocking scheme learning approach which incorporates active learning techniques [37; 49] into the blocking scheme learning process. With a limited label budget, our approach can learn a blocking scheme to generate high quality blocks w.r.t. a pre-defined error rate such as pair completeness or pair quality. Two strategies called *active sampling* and *active branching* are proposed to select samples and generate blocking schemes actively and efficiently. Specifically, given a set of blocking predicates, the active sampling strategy aims to alleviate the class imbalance problem of entity resolution by selecting informative training samples w.r.t. different combination of blocking predicates; the active branching strategy determines whether a further combination (in either conjunction or disjunction form) of blocking predicates should be generated. Besides, compared with existing work, our approach can generate blocking schemes with the conjunctions or disjunctions of an arbitrary number of blocking predicates, instead of limiting at most k predicates to be used in conjunctions [11; 84]. We experimentally verify that our approach yields high quality blocks within a specified error rate and a limited budget of labels, and it outperforms several baseline approaches including the state-of-the-art approaches over four real-world datasets.

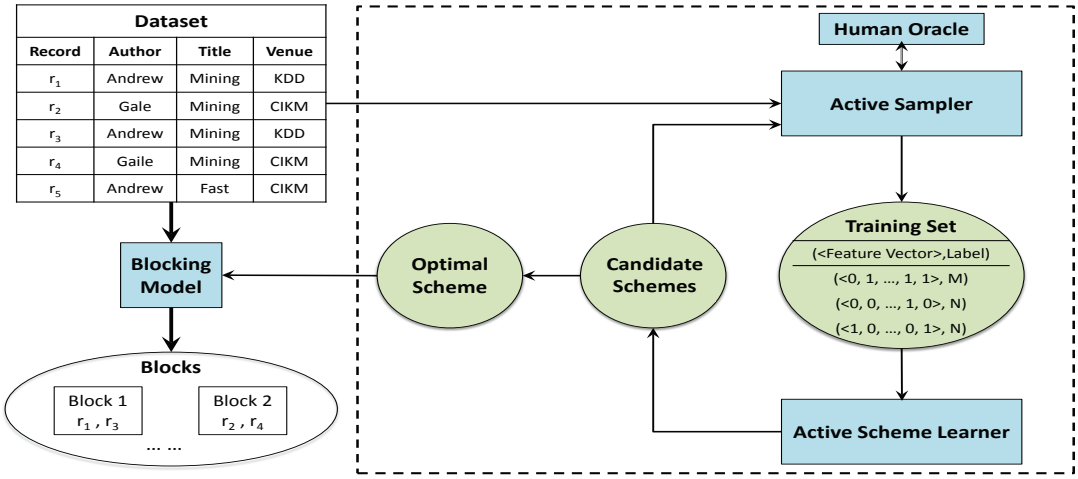

Figure 4.1: Overview of the active blocking scheme learning approach.

Fig. 4.1 illustrates our proposed approach, which works as follows: Given a dataset R, an active sampler selects samples from R based on a set of candidate schemes, and asks a human oracle for labels. Then an active scheme learner generates a set of refined candidate schemes, enabling the active sampler to adaptively select more samples. Within a limited label budget and an error rate, an optimal scheme will be selected from the candidate schemes.

The rest of this chapter is structured as follows. We first formulate our learning task as an active scheme learning problem in Section 4.2. Two strategies proposed to solve the problem,

together with our active scheme learning algorithm ASL, are presented in Section 4.3. In Section 4.4, we theoretically analyze our approach in both search complexity and time complexity. Some experimental results are shown in Section 4.5. This chapter is concluded in Section 4.6.

4.2 Problem Formulation

We formulate our blocking scheme learning task as follows:

Definition 1. *Given a human oracle ζ, and an error rate $\epsilon \in [0, 1]$, the **active scheme learning problem** is to learn a blocking scheme s in terms of the following objective function, through actively selecting a training set T:*

$$\begin{aligned} &\textbf{\textit{minimize}} \quad |fp(B_s)| \\ &\textbf{\textit{subject to}} \quad \frac{|fn(B_s)|}{|tp(B_s)|} \leq \epsilon, \text{ and } |T| \leq budget(\zeta) \end{aligned} \tag{4.1}$$

4.3 Active Scheme Learning Framework

In our active scheme learning framework, we develop two complementary and integrated strategies to adaptively generate a set of blocking schemes and learn the optimal one based on actively selected samples. These two strategies are introduced in Section 4.3.1 and Section 4.3.2, respectively. The algorithm we propose is called *Active Scheme Learning (ASL)* and described in Section 4.3.3.

4.3.1 Active Sampling

To deal with the active scheme learning problem, we need both match and non-match samples for training. However, one of the well-known challenges in entity resolution is the class imbalance problem [149]. That is, if samples are selected randomly, there are usually much more non-matches than matches.

Previously, it has been reported that training data with imbalanced classes has impeded the performance of learning approaches in entity resolution [11; 17; 84]. Although the ratio of non-matches and matches may vary in different datasets, using random sampling can almost always generate much more non-matches than matches in training data, as illustrated in Fig. 4.2. With random sampling, there are far more non-matches than matches selected w.r.t. the similarity of attributes title and authors in the dataset Cora. However, samples selected using our active sampling strategy contain more matches compared with random sampling. To effectively learn blocking schemes, it is thus important to have a training set where matches and non-matches are balanced.

In the following, we propose an active sampling strategy, aiming to generate a balanced training set by using only a small amount of labels. We observe that the correlation between the similarity of features (i.e., how similar the features of two records are) and the labels (i.e., $\{M, N\}$) can be leveraged to reduce the imbalance of matches and non-matches in sampling.

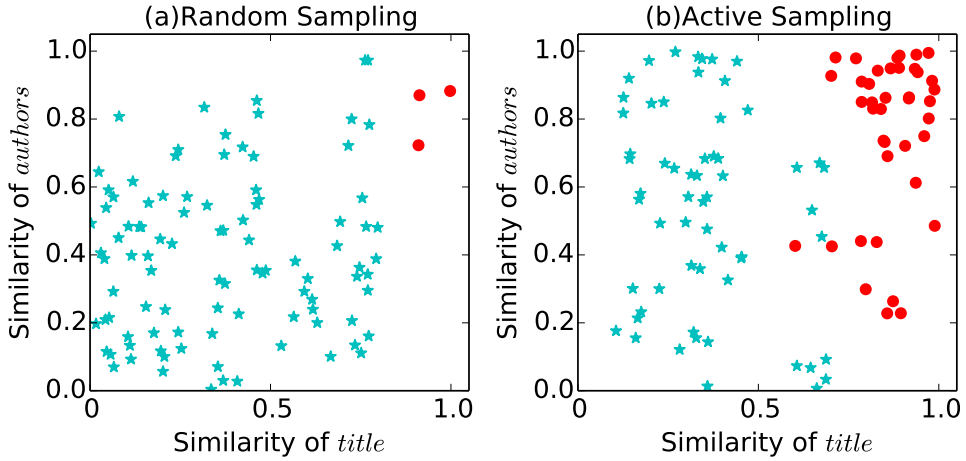

Figure 4.2: A comparison on the sample distribution of 100 samples from Cora dataset: (a) random sampling, and (b) active sampling, where a red circle indicates a matched sample and a blue star indicates a non-matched sample.

More specifically, this observation is based on two empirical facts: (1) the more similar two records are, the higher probability they can be a match, as reported in previous works [3; 8]; (2) the similarity of some features may correlate more closely with matches than the similarity of other features. For example, in a bibliographic dataset, if we have 10 record pairs whose title are the same and another 10 record pairs whose publication year are the same, then it is likely that the former pairs refer to more matches in comparison with the latter pairs. Hence, by virtue of the correlation between the similarity of features and the match and non-match labels, a balanced set of similar and dissimilar records likely represents a training set with balanced matches and non-matches. To formulate this observation, we define the notion of balance rate in the following.

Definition 2. *Let s be a blocking scheme and X a non-empty sample set of blocking vectors, the **balance rate** of X in terms of s, denoted as $\beta(s, X)$, is defined as:*

$$\beta(s, X) = \frac{|\{x_i \in X | s(x_i) = true\}| - |\{x_i \in X | s(x_i) = false\}|}{|X|} \tag{4.2}$$

Conceptually, the balance rate describes how balance or imbalance of the samples in X by comparing the number of similar samples to that of dissimilar samples in terms of a given blocking scheme s. The range of balance rate is $[-1, 1]$. If $\beta(s, X) = 1$, there are all similar samples in X with regard to s, whereas $\beta(s, X) = -1$ means all the samples are dissimilar samples. In these two cases, X is highly imbalanced. If $\beta(s, X) = 0$, there is an equal number of similar and dissimilar samples, indicating that X is balanced.

Based on the notion of balance rate, we convert the class imbalance problem into the balanced sampling problem as follows:

Definition 3. *Given a set of blocking scheme S and a label budget $n \leq budget(\zeta)$, the **balanced sampling problem** is to select a training set $T = (X, Y)$, where $|X| = n$, in order to:*

$$\textbf{\textit{minimize}} \sum_{s_i \in S} \beta(s_i, X)^2 \tag{4.3}$$

For two different blocking schemes s_1 and s_2, they may have different balance rates over the same sample set X, i.e., $\beta(s_1, X) \neq \beta(s_2, X)$ is possible. The objective here is to find a training set that minimizes the balance rates in terms of the given set of blocking schemes. This process is called *Active Sampling*. The optimal case is $\beta(s_i, X) = 0, \forall s_i \in S$ according to the objective function above, but sometimes it is impossible to achieve this in real world applications.

Example 4.3.1. *Consider the dataset example in Table 2.1 again, and let X refer to the set of all possible pairs of records from this table. Then, for a blocking scheme $s_1 = \langle$Title, Same-value$\rangle$, there are three similar samples out of all possible samples: $\langle r_1, r_2\rangle$, $\langle r_1, r_4\rangle$ and $\langle r_2, r_4\rangle$. That is, $\beta(s_1, X) = 0.4$. For another blocking scheme $s_2 = \langle$Authors, Same-soundex$\rangle$ described in Example 2.1.1, we have four similar samples: $\langle r_1, r_2\rangle$, $\langle r_1, r_3\rangle$, $\langle r_2, r_3\rangle$ and $\langle r_4, r_5\rangle$, i.e., $\beta(s_2, X) = 0.2$.*

4.3.2 Active Branching

Given n blocking predicates, we have 2^n possible blocking schemes which can be constructed upon n blocking predicates in the form of only conjunctions or disjunctions. Thus, the number of all possible blocking schemes which can be constructed through arbitrary combinations of conjunction and disjunction of blocking predicates is more than 2^n. To efficiently learn blocking schemes, we therefore propose a hierarchical blocking scheme learning strategy called *active branching* to avoid enumerating all possible blocking schemes and reduce the number of candidate blocking schemes to $\frac{n(n+1)}{2}$.

For a blocking scheme s, there are two types of branches through which we can extend s with another blocking predicate: conjunction and disjunction. Let s_1 and s_2 be two blocking schemes. We have the following lemmas.

Lemma 4.3.1. *For the conjunction of s_1 and s_2, the following holds:*

$$|fp(B_{s_i})| \geq |fp(B_{s_1 \wedge s_2})|, \text{ where } i = 1, 2 \tag{4.4}$$

Proof. For any true negative record pair $tn \notin B_{s_1}$, we have $tn \notin B_{s_1 \wedge s_2}$, which means $|tn(B_{s_1})| \leq |tn(B_{s_1 \wedge s_2})|$. Since the sum of true negatives and false positives is a constant for a given dataset, we have $|fp(B_{s_1})| \geq |fp(B_{s_1 \wedge s_2})|$. □ □

Lemma 4.3.2. *For the disjunction of s_1 and s_2, the following holds:*

$$\frac{|fn(B_{s_i})|}{|tp(B_{s_i})|} \geq \frac{|fn(B_{s_1 \vee s_2})|}{|tp(B_{s_1 \vee s_2})|}, \text{ where } i = 1, 2 \tag{4.5}$$

Proof. For any true positive record pair $tp \in B_{s_1}$, we have $tp \in B_{s_1} \cup B_{s_2} = B_{s_1 \vee s_2}$. This is, the number of true positives generated by s_1 cannot be larger than that generated by $s_1 \vee s_2$,

i.e., $|tp(B_{s_1})| \leq |tp(B_{s_1 \vee s_2})|$. Since the sum of true positives and false negatives is constant, we have $|fn(B_{s_1})| \geq |fn(B_{s_1 \vee s_2})|$. □ □

Based on Lemmas 4.3.1 and 4.3.2, we develop an active branching strategy as follows. First, a locally optimal blocking scheme is learned from a set of candidate schemes. Then, by Lemma 4.3.1, the locally optimal blocking scheme is extended. If no locally optimal blocking scheme is learned, according to Lemma 4.3.2, the active branching strategy selects the one with minimal error rate and extends it in disjunction with other blocking predicates to reduce the error rate. The extended blocking schemes are then used as a set of candidate schemes for active sampling to select more samples. Based on more samples, the active branching strategy adaptively refines the locally optimal scheme. This process iterates until the label budget is used out.

4.3.3 Algorithm Description

We present the details for our proposed algorithm called *Active Scheme Learning (ASL)* used in our framework. A high-level description is shown in Algorithm 1. Let S be a set of blocking schemes, where each blocking scheme $s_i \in S$ is a blocking predicate at the beginning. The budget usage is initially set to zero, i.e., $n = 0$. A set of blocking vectors is selected from the dataset as seed samples (lines 1 and 2).

After initialization, the algorithm iterates until the number of samples in the training set reaches the budget limit (line 3). At the beginning of each iteration, the active sampling strategy is applied to generate a training set (lines 4 to 10). For each blocking scheme $s_i \in S$, the samples are selected in two steps: Firstly, the balance rate of this blocking scheme s_i is calculated (lines 5 and 7). Secondly, a blocking vector to reduce this balance rate is selected from the dataset (lines 6 and 8). Then the samples are labeled by the human oracle and stored in the training set T. The usage of label budget is increased, accordingly (lines 9 and 10).

A locally optimal blocking scheme s is searched among a set of blocking schemes S over the training set, according to a specified error rate ϵ (line 11). If the blocking scheme s is found, new blocking schemes are generated by extending s to a conjunction with each of the blocking schemes in S_{prev} (lines 12 and 13). Otherwise a blocking scheme with the minimal error rate is selected and new schemes are generated using disjunctions (lines 14 to 16).

4.4 Theoretical Analysis

In this section, we discuss the search complexity and the time complexity for learning blocking schemes.

Search complexity. We first discuss about the search complexity of the optimal blocking schemes among the number of all possible blocking schemes compared with our algorithm w.r.t. a given number of blocking predicates. We have at most 2^n blocking schemes composed of n blocking predicates in conjunctions. Furthermore, if a blocking scheme is composed of at most n different blocking predicates (i.e. n-ary blocking scheme) in disjunction of conjunctions, we can regard them as the *monotonic boolean functions*, which are defined to be

Algorithm 1: Active Scheme Learning (ASL)

Input: Dataset: R
Error rate $\epsilon \in [0, 1]$
Human oracle ζ
Set of blocking predicates P
Sample size k

Output: A blocking scheme s

$S = S_{prev} = P$, $n = 0$, $T = \varnothing$, $X = \varnothing$
$X = X \cup \text{RANDOM\_SAMPLE}(R)$
while $n < budget(\zeta)$ **do**
  for *each* $s_i \in S$ **do** `// Begin active sampling`
    if $\beta(s_i, X) \leq 0$ **then**
      $X = X \cup \text{SIMILAR\_SAMPLE}(R, s_i, k)$
    else
      $X = X \cup \text{DISSIMILAR\_SAMPLE}(R, s_i, k)$
    $n = |X|$ `// End active sampling`
  $T = T \cup \{(x_i, \zeta(x_i)) | x_i \in X\}$ `// Add labeled samples into` T
  $s = \text{FIND\_OPTIMAL\_SCHEME}(S, T, \epsilon)$; $S_{prev} = S$ `// Begin active branching`
  if $\text{FOUND}(s)$ **then**
    $S = \{s \wedge s_i | s_i \in S_{prev}\}$
  else
    $s = \text{FIND\_APPROXIMATE\_SCHEME}(S, T, \epsilon)$
    $S = \{s \vee s_i | s_i \in S_{prev}\}$ `// End active branching`
Return s

expressions combining the inputs (which may appear more than once) using only the operators conjunction and disjunction (in particular "not" is forbidden) [91]. Hence, the searching complexity for all possible blocking schemes can be $O(2^{\binom{n}{\lfloor n/2 \rfloor}})$ asymptotically, which is also known as the *Dedekind Number*. Learning a blocking scheme by considering all possible blocking schemes may yield high accuracy. However, it would be space-consuming and label-inefficient, especially when the number of blocking predicates is large.

For our algorithm, given n blocking predicates with a sufficient label budget, in the worst case, we can learn a n-ary blocking scheme as output. During the learning process, we first need to search for all n blocking schemes. Then based on the locally optimal one, we need to search for $n-1$ blocking schemes of 2-ary, and then, $n-2$ blocking schemes of 3-ary. Accordingly, the search complexity of our algorithm is $O(n^2)$.

Time complexity. We further discuss the time complexity of sampling in our algorithm. To tackle the class imbalance problem, we select both similar and dissimilar samples w.r.t. a given blocking scheme s. However, as explained in Section 4.3.1, there may exist a high imbalance ratio between similar and dissimilar samples. In the worst case, we have to traverse the whole dataset to obtain one sample. Hence the time complexity to generate k samples is $O(|R| \times k)$ for one blocking predicate, where R is a dataset and k is the sample size for the blocking scheme learning.

To avoid traversing an entire dataset to find samples (in the worst case) and make our algorithm efficient under a large k, we have implemented index tables for traditional blocking predicates as defined in the previous works [11; 84]. This implementation helps the algorithm to easily select the samples it needs and choose a similar or dissimilar sample in linear time instead of traversing the dataset under a given blocking predicate. Therefore, the time complexity is reduced to $O(|R| + k)$ for a blocking predicate.

Example 4.4.1. *Let us consider* $\langle$*Authors, Same-soundex*$\rangle$ *which is a blocking predicate previously discussed in Example 2.1.1. We can place all records into at most* 26×7 *buckets, where each bucket represents a soundex value such as G4. Hence, a match can be easily selected from the same bucket or a non-match can be selected from two different buckets. In this case, the time complexity is* $O(|R| + k)$.

In our work, block predicates are chosen by users. If a user chooses similarity-based blocking predicates, where the records need to be compared in pairs, and thus the index value for a single record is not unique, then the time complexity of selecting k samples will remain $|R| \times k$ for such a blocking predicate.

4.5 Experiments

We have conducted experiments to empirically verify our skyline learning algorithms, aiming to answer the following questions:

(1) How do the error rate ϵ and the label budget affect the learning results in our approach?

(2) What are the accuracy and efficiency of our active scheme learning approach compared with the state-of-the-art approaches?

4.5.1 Experimental Setup

Datasets. We have used four datasets in the experiments: (1) *Cora*, *DBLP-Scholar*, *DBLP-ACM* and *North Carolina Voter Registration (NCVoter)*. The characteristics of these data sets are summarized in Table 2.2.

Blocking functions. Blocking functions are widely used in scheme-based blocking approaches. For example, a person called Michael always writes his name as Mike, and a person called Gaile is recorded as Gale by the phone since they have similar pronunciation. When the name-related attribute is applied for the blocking scheme to decide whether two records are a positive pair or not, we have no knowledge about if such cases will present. In order to avoid classifying true match pairs as negative ones, blocking functions, each of which is associated with an attribute, are used in blocking schemes. We have used the following blocking functions to generate blocking predicates in our experiments:

- **Exact-match:** This function regards each pair of input attribute values as strings, and compares the characters one by one, if all the characters are the same, i.e., the pair of strings is exactly match, the function will return an "1" as match, otherwise it returns a "0" as non-match.
- **Soundex:** Soundex is one of the best-known and widely used phonetic encoding functions, which converts each attribute value (a string) into codes made of one letter and several digits. The main steps are as below: (1) Keep the first letter of a string. (2) Remove all the following characters from the string: a, e, i, o, u, y, h, w. (3) Replace all consonants from position 2 onwards with digits using these rules: b, f, p, v $\rightarrow$ 1; c, g, j, k, q, s, x, z $\rightarrow$ 2; d, t $\rightarrow$ 3; l $\rightarrow$ 4; m, n $\rightarrow$ 5; r $\rightarrow$ 6. Hence we have several digits transferred from the letters. Furthermore, we have defined the length of the converted code as 4, i.e., if length of a code is less than 4, we add zeros; if it is longer, we truncate it at length 4.
- **Double-Metaphone:** Even Soundex is widely used, a major drawback is that it is specifically aimed at English names when it was designed. Therefore, it is not suitable for datasets that contain names from different languages. The Double-Metaphone function attempts to accomplish this by providing two possible codes instead of one for each input string. For example, the Polish name "Kuczewski" will be encoded as "kssk" and " kxfsk".
- **Substring:** Substring function aims to convert a pair of input strings into the specific (e.g. first two characters, last three characters) substrings, and compare the characters one by one contained these substrings as the process of Exact-match.

These blocking functions have been applied to all attributes in the datasets depicted in Table 2.3, which accordingly leads to 16 or 72 different blocking predicates for datasets as shown in Table 2.2.

Baseline approaches. Since no active learning approaches have been proposed on blocking scheme learning, we have compared our approach (ASL) with the following three baseline approaches:

Table 4.1: Comparison on label cost by ASL and RSL over four real datasets.

Error rate	Cora	DBLP-Scholar	DBLP-ACM	NCVoter
0.8	600	500	300	300
0.6	**400**	350	200	350
0.4	450	**250**	**150**	250
0.2	550	300	200	**200**
0.1	500	**250**	300	250
RSL	8,000	10,000+	2,500	10,000+

- **Fisher [84]:** this is the state-of-the-art unsupervised scheme learning approach proposed by Kejriwal and Miranker. Details of this approach have been outlined in Chapter 3.
- **TBlo [47]:** this is a traditional blocking approach based on expert-selected attributes. In the survey [27], this approach has a better performance than the other approaches in terms of the F-measure results.
- **RSL (Random Scheme Learning):** this approach uses random sampling, instead of active sampling, to build the training set and learn blocking schemes. We run the RSL ten times, and present the average results of blocking schemes it learned.

Measurements. *Reduction Ratio (RR)*, *Pairs Completeness (PC)*, *Pairs Quality (PQ)* and *F-measure (FM)* are widely used in entity resolution, which are introduced in Section 2.2.2. Additionally, *constraint satisfaction* $CS = \frac{N_s}{N}$ is defined to show the stability of our approaches under different hyper-parameters, e.g. error rates, where N_s is the times of learning an optimal blocking scheme by an algorithm and N is the total times the algorithm runs.

4.5.2 Results and Discussion

Now we present our experimental results in terms of the label cost, constraint satisfaction, blocking quality and blocking efficiency.

4.5.2.1 Label Efficiency

Label cost. In order to compare the label cost required by ASL and RSL for achieving the same block quality, we present the numbers of labels needed by our approach to generate a blocking scheme with CS=100% under different error rates, and compare them with the labels required by RSL in Table 4.1. In our experiments, the label budget for ASL under a given error rate starts with 50, and then increases by 50. The label budget for RSL starts with 500, and increases by 500 each time. Both ASL and RSL algorithms terminate when the learned blocking schemes remain the same in ten consecutive runs. We can see that, running our ASL approach over different datasets need different label cost to achieve a certain error rate. For a specific dataset, the minimum label cost varies w.r.t. different error rate. However,

Observation 1. *Our ASL approach using active sampling strategy can use significantly less labels compared with random sampling strategy under different error rates.*

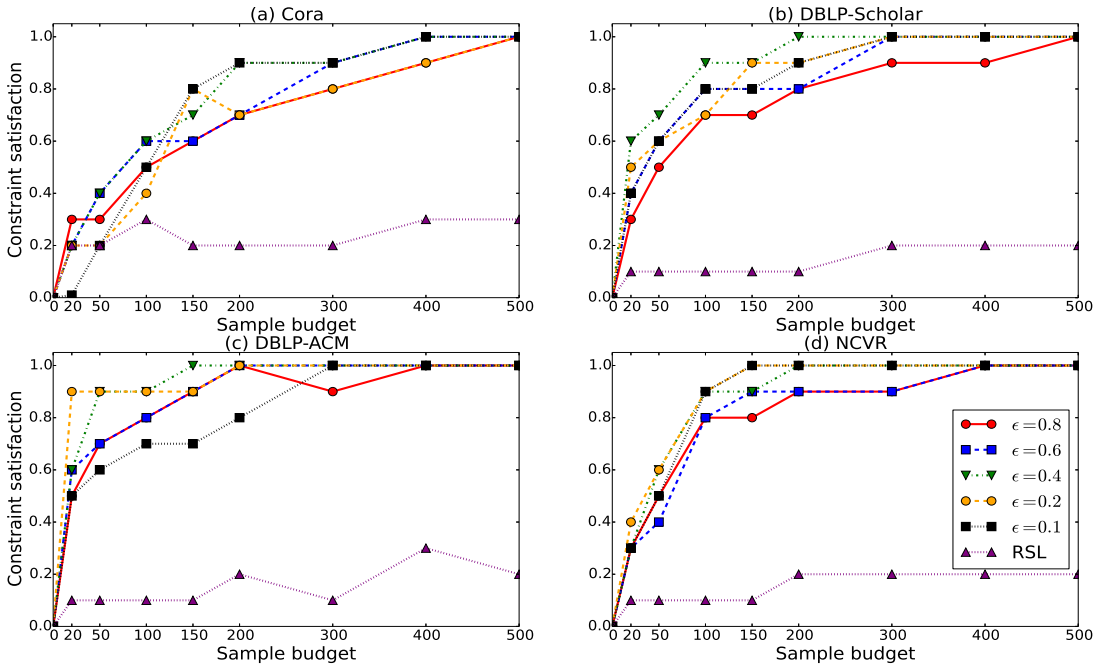

Figure 4.3: Comparison on constraint satisfaction by Active Scheme Learning (ASL) and Random Scheme Learning (RSL) under different label budgets and different error rates over four datasets.

Constraint satisfaction. We have conducted experiments to evaluate the constraint satisfaction. In Fig. 4.3, the results are presented under different error rates $\epsilon \in \{0.1, 0.2, 0.4, 0.6, 0.8\}$ and different label budgets ranging from 20 to 500 over four real datasets. We use the total label budget as the training label size for RSL to make a fair comparison on active sampling and random sampling. Our experimental results show that random sampling with a limited label sizes often fails to produce an optimal blocking scheme. Additionally, both error rate and label budget can affect the constraint satisfaction. As shown in Fig. 4.3(a)-(d), when the label budget increases, the CS value goes up. In general, when ϵ becomes lower, the CS value decreases. This is because a lower error rate is usually harder to achieve, and thus no scheme that satisfies the error rate can be learned in some cases. However, if the error rate is set too high (e.g. the red line), it could generate a large number of blocking schemes satisfying the error rate, and the learned blocking scheme may vary depending on the training set.

Observation 2. *Our ASL approach with active sampling strategy can use much less labels to achieve the same or even better constraint satisfaction compared with random sampling strategy. However, the best performance varies w.r.t. the datasets and the PC thresholds.*

4.5.2.2 Blocking Quality

We present the experimental results of four measures (i.e. RR, PC, PQ, and FM) for our approach and the baseline approaches. In Fig. 4.4(a), all the approaches yield high RR values

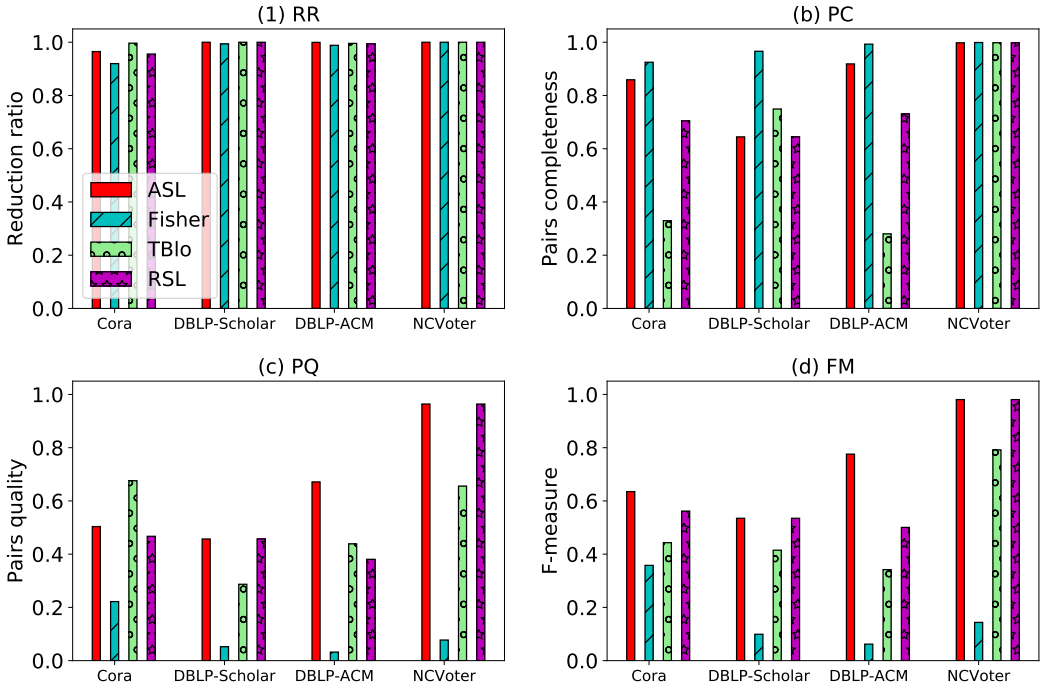

Figure 4.4: Comparison on blocking quality by different blocking approaches over four datasets using the measures: (a) RR, (b) PC, (c) PQ, and (d) FM.

Table 4.2: **Comparison on the number of record pairs generated by different approaches.**

	TBlo	Fisher	ASL	RSL
Cora	**2,945**	67,290	29,306	17,974
DBLP-Scholar	6,163	1,039,242	**3,328**	**3,328**
DBLP-ACM	25,279	69,037	**3,043**	17,446
NCVoter	932,239	7,902,910	**634,121**	**634,121**

over four datasets. In Fig. 4.4(b), the PC values of our approach are not the highest over the four datasets, but they are not much lower than the highest one (i.e. within 10% lower except in DBLP-Scholar). However, out approach can generate higher PQ values than all the other approaches, from 15% higher in NCVoter (0.9956 vs 0.8655) to 20 times higher in DBLP-ACM (0.6714 vs 0.0320), as shown in Fig. 4.4(c). The FM results are shown in Fig. 4.4(d).

Observation 3. *Our ASL approach outperforms all the baselines over all the datasets w.r.t. RR and FM, and has a trade-off between PC and PQ.*

4.5.2.3 Blocking Efficiency

Since blocking aims to reduce the number of pairs to be compared in entity resolution, we evaluate the efficiency of blocking schemes by the number of record pairs each approach generates. As shown in Table 4.2, *TBlo* generates the minimal number of record pairs in Cora. This is due to the scheme that is manually selected by domain experts. *Fisher* targeted to learn disjunctive schemes, which can lead to large blocks, thus the number of record pairs is the largest over four datasets. *ASL* considers a trade-off between PC and PQ, and the number of record pairs is often small. In *RSL*, we use the same label size as *ASL*; thus it may learn a blocking scheme that is different from the one learned by RSL, and accordingly generates different numbers of record pairs for some datasets such as Cora and DBLP-ACM. When a sufficient number of samples is used, the results of *ASL* and *RSL* would be close.

Observation 4. *Our ASL approach highly reduces the non-matches for blocking while keeps high pair quality and pair completeness at the same time.*

4.6 Summary

In this chapter, we have used active learning techniques to develop a blocking scheme learning approach. Our approach overcomes the weaknesses of existing works in two aspects: (1) Previously, supervised blocking scheme learning approaches require a large number of labels for learning a blocking scheme, which is an expensive task for entity resolution; (2) Existing unsupervised blocking scheme learning approaches generate training sets based on the similarity of record pairs, instead of their true labels, thus the training quality can not be guaranteed. Our experimental results show that the proposed approach outperforms the baseline approaches under a specific error rate with a sample budget.

Chapter 5

Skyblocking for Entity Resolution

5.1 Introduction

As we have shown, Active Scheme Learning (ASL) approach can learn the optimal blocking scheme with only a small number of labeled samples. However, a proper error rate may not be able to obtain in real world scenarios. Additionally, one may prefer different criteria for blocking, such as PC and PQ.

Consequently, the question that naturally arises is: can we efficiently identify all possible most preferred blocking schemes in terms of a given set of selected criteria? To answer this question, we may consider, one blocking scheme is preferred than another if it has as good or better performance in all criteria (e.g. PC, PQ) and has better performance in at least one criterion. Hence we propose a scheme skyline learning framework for blocking, called *skyblocking*, which incorporates skyline techniques into an active learning process of blocking schemes on the skyline. The target of skyblocking is to find a range of blocking schemes (as shown in Fig. 5.1) under different user preferences and within a limited label budget, and each of such blocking schemes is optimal with respect to one or more selection criteria.

Traditionally, skyline techniques have been widely used for database queries, called skyline queries [80; 21; 131]. The result of a skyline query consists of multi-dimensional points for which there is no one point having better values in all the dimensions [20]. However, applying skyline querying technique to learning scheme skylines is a non-trivial task. Different from skyline queries in a database [21], in which the dimensions of a skyline correspond to attributes and the points correspond to records, scheme skylines in our work have the dimensions corresponding to selection criteria for blocking, and points corresponding to blocking schemes in this multi-dimensional space, called *scheme space*. Learning scheme skylines becomes challenging, due to the following unique characteristics of blocking schemes.

- It is hard to obtain labels in real-world entity resolution applications. Particularly, it is well-known that match and non-match labels for entity resolution are often highly imbalanced, called *the class imbalance problem* [49; 29; 149]. Although active learning techniques help to reduce the label cost in learning one blocking scheme w.r.t. ASL [135], far more labels are needed if we want to learn all the blocking schemes to draw the scheme skyline. Additionally, to find scheme skylines effectively, we need a training set that contains a sufficient number of matches. Hence, the first challenge we must address is how to select more matched samples within a limited label budget for blocking scheme

Blocking scheme	PC	PQ
s_1	0.13	0.76
s_2	0.31	0.99
s_3	0.58	0.76
s_4	0.84	0.40
s_5	0.86	0.50
...	...	...

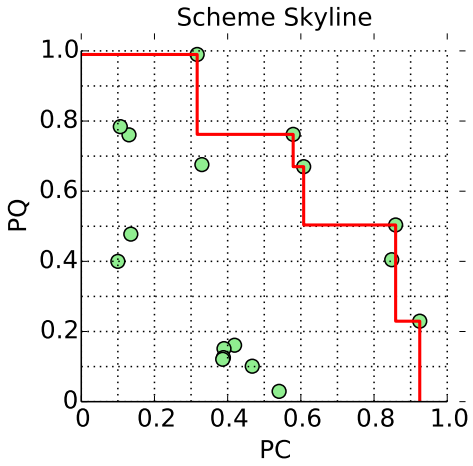

Figure 5.1: An example for scheme skyline where schemes on the skyline refer to the points in the red line. The blocking schemes (green points) are presented in a 2-dimensional space of PC and PQ, and their corresponding values are shown in the table (left).

learning in a scheme space.

- A scheme space can be very large, making an exhaustive search for blocking schemes on a skyline computationally expensive. If a blocking scheme is composed of at most n different blocking predicates, the number of all possible blocking schemes can be $O(2^{\binom{n}{\lfloor n/2 \rfloor}})$ asymptotically. Nevertheless, only a small portion of these blocking schemes are on a skyline. Thus, the second challenge we need to act on is how to design efficient skyline algorithms to explore a large scheme space efficiently for finding blocking schemes on a skyline. Enumerating through the entire scheme space by checking a blocking scheme each time is obviously not efficient in practice, and even infeasible for large datasets with hundreds or thousands of attributes.

To overcome the above challenges, in this work, we formulate a novel scheme skyline learning problem for entity resolution, which hierarchically learns a scheme skyline based on blocking predicates to avoid enumerating all possible blocking schemes. Solving this problem would lead to generating a range of optimal blocking schemes with respect to different blocking criteria and thus enable users to choose their preferred blocking schemes. We propose three novel scheme skyline learning algorithms to efficiently learn scheme skylines within a limited label budget. In these algorithms, we tackle the class imbalance problem by first converting this problem into the balanced sampling problem and then developing an active sampling strategy to actively select informative samples. We also develop a scheme extension strategy for efficiently identifying schemes that are possibly on a skyline in order to reduce the search space and label cost used in the learning process. We have evaluated the efficiency and effectiveness of our scheme skyline learning algorithms over five real-world datasets. The experimental results show that our algorithms outperform the baseline approaches in all of the following aspects: label efficiency, time efficiency and blocking quality.

The rest of this chapter is structured as follows. Section 5.2 introduces the notations used

in the chapter and how we formulate the scheme skyline learning under a limited number of labels. Three skyline algorithms for learning scheme skylines are presented in Section 5.3, and then Section 5.4 analyzes the search complexity and the time complexity of our algorithms. Section 5.5 discusses our experimental results. We conclude the chapter in Section 5.6.

5.2 Problem Formulation

Given a dataset R and a set of blocking schemes S over R, a *scheme space* is a d-dimensional space consisting of points in $[0,1]^d$. Each point, denoted as $p(s)$, is associated with a blocking scheme $s \in S$ such that $p(s) = (p_1.s, p_2.s, \ldots, p_d.s)$, where each $p_i.s$ indicates the value of s in the i-th dimension of this scheme space and $i \in [1,d]$. For example, in Fig. 5.1, each green point is associated with a blocking scheme in a 2-dimensional space.

Definition 4. *(*Dominance*) Given two blocking schemes s_1 and s_2, we say s_1* dominates *s_2, denoted as $s_1 \succ s_2$, iff $\forall i \in [1,d], p_i.s_1 \geq p_i.s_2$ and $\exists j \in [1,d], p_j.s_1 > p_j.s_2$.*

Here, $p_i.s_1 \geq p_i.s_2$ indicates that the value of s_1 is larger than that of s_2 in the i-th dimension. Based on the notion of dominance between two blocking schemes, we define the notion of scheme skyline.

Definition 5. *(*Scheme skyline*) Given a dataset R and a set of blocking schemes S, a* scheme skyline *S^* over S is a subset of S, where each scheme $s \in S^*$ is not dominated by any other blocking scheme in S, i.e., $S^* = \{s \in S : \nexists s' \in (S - S^*), s' \succ s\}$.*

Without loss of generality, we will discuss a scheme space with $d = 2$ in the rest of this chapter: one dimension indicates the PC values of blocking schemes and the other dimension indicates the PQ values of blocking schemes. Note that, it is possible to extend the dimensions of a scheme space by taking into account other measures for blocking schemes, such as reduction ratio (RR) and run time (RT) [94]. We will further discuss the properties of such measures later in Section 5.3.

We now define the problem of scheme skyline learning, which allows us to select desired blocking schemes from a scheme space depending upon which blocking criteria we prefer.

Definition 6. *(*Scheme skyline learning problem*) Let ζ be a human oracle, R be a dataset and S be a set of blocking schemes over R. Then the* scheme skyline learning problem *is to learn a scheme skyline S^* over S through actively selecting a training set T from R, such that $|T| \leq budget(\zeta)$ holds.*

5.3 Scheme Skyline Learning Framework

We now propose three different algorithms to solve the scheme skyline learning problem. To tackle the class imbalance problem under a limited number of labeled samples, we adopt the two strategies introduced in Chapter 4, to make efficient usage of samples. Following the active sampling strategy defined in Definition 2 and Definition 3, we can build a nearly balanced sample set. Additionally, since one of the key challenges faced by scheme skyline learning is

to efficiently search for blocking schemes in a skyline from a large scheme space, we will first discuss an efficient scheme extension strategy based on Section 4.3.2. Then, we will introduce three skyline learning algorithms, and discuss their advantages and disadvantages. Finally, we will formally analyze the search and time complexity of our skyline learning algorithms.

5.3.1 Scheme Extension Strategy

Given a blocking scheme s, there are two kinds of possible extensions: conjunction and disjunction. Although iterating all the possible extensions is theoretically feasible, it is not efficient. Hence, one would expect that, if a conjunction or disjunction can be decided before extension, the searching space would be at least reduced by half. In line with this, we explore the monotonic property of measures for blocking. Let s_i and s_j be two blocking schemes. We have the following lemma.

Lemma 5.3.1.

$$PC(s_i) \leq PC(s_i \vee s_j) \tag{5.1}$$

$$PC(s_i) \geq PC(s_i \wedge s_j) \tag{5.2}$$

Proof. Suppose that x is a true positive in B_{s_i}, then we would have $x \in B_{s_i} \cup B_{s_j} = B_{(s_i \vee s_j)}$. This is to say, true positives generated by either s_i or s_j must also be generated by $s_i \vee s_j$, and must contain all true positives generated by $s_i \wedge s_j$. Since we know the sum of true positives and false negatives is constant, which refers to the total number of matched pairs in a dataset, by the definition of PC, the lemma is proven.

□

This lemma implies that, if a scheme generates blocks with a low PC value, we can extend it with disjunction in order to increase the PC value. On the contrary, if extending this scheme by conjunction, the PC value will be decreased.

Generally, if a measure of blocking [118] has a monotonic property in terms of the conjunction and disjunction of blocking schemes, then it can be used as a dimension for scheme skylines in a scheme space and our scheme extension strategy can be applied. For example, reduction ratio (RR) has the following monotonic property, which is opposite to the monotomic property of PC, i.e., $RR(s_i) \geq RR(s_i \vee s_j)$ and $RR(s_i) \leq RR(s_i \wedge s_j)$. Consequently, an extension of conjunction helps increase RR values, but an extension of a disjunction can lead to lower RR values than before. Similarly, run time (RT) also has a monotonic property because it is monotonously increasing in terms of extensions, regardless whether an extension is a conjunction or a distinction, i.e., $RT(s_i) \leq RT(s_i \vee s_j)$ and $RT(s_i) \leq RT(s_i \wedge s_j)$.

5.3.2 Naive Skyline Learning

A naive way to learn skyline blocking schemes is to learn optimal blocking schemes under different thresholds in all dimensions. Then, based on these optimal blocking schemes, a scheme skyline can be obtained. In the following, we formulate the notion of optimal blocking scheme.

Definition 7. *(*Optimal blocking scheme*) Given a human oracle ζ, and a PC threshold $\epsilon \in [0,1]$, an* optimal blocking scheme *is a blocking scheme $s_{\epsilon,\zeta}$ w.r.t. the following objective function, through selecting a training set T:*

$$\begin{aligned} &\textbf{\textit{maximize}} \quad PQ \\ &\textbf{\textit{subject to}} \quad PC \geq \epsilon, \text{ and } |T| \leq budget(\zeta) \end{aligned} \tag{5.3}$$

Let S be a subset of all possible blocking schemes. Then a blocking scheme s is called a *locally optimal blocking scheme* from S if $s \in S$ and it satisfies Eq. 5.3. Algorithm 2 describes a procedure of learning locally optimal blocking schemes using the active sampling strategy and the scheme extension strategy discussed in Section 5.3.1.

Algorithm 2: Active Scheme Learning Plus(ASL+) Procedure

Input: Dataset: R
PC threshold $\epsilon \in (0,1]$
Human oracle ζ with $budget(\zeta)$
Set of blocking predicates P
Sample size k
Output: A blocking scheme s

```
1  S = S_prev = P, n = 0, T = ∅, X = ∅
2  X = X ∪ RANDOM_SAMPLE(R)
3  while n < budget(ζ) do
4      for each s_i ∈ S do
5          if β(s_i, X) ≤ 0 then
6              X = X ∪ SIMILAR_SAMPLE(R, s_i, k)
7          else
8              X = X ∪ DISSIMILAR_SAMPLE(R, s_i, k)
9          n = |X|
10     T = T ∪ {(x_i, ζ(x_i)) | x_i ∈ X}
11     s = FIND_OPTIMAL_SCHEME(S, T, ε); S_prev = S
12     if FOUND(s) then
13         S = {s ∧ s_i | s_i ∈ S_prev}
14     else
15         s = FIND_APPROXIMATE_SCHEME(S, T, ε)
16         S = {s ∨ s_i | s_i ∈ S_prev}
17 Return s
```

This algorithm is designed based on Algorithm 1 introduced in Chapter 4, and here we call it the *Active Scheme Learning Plus* (ASL+) procedure. Different from that in Chapter 4, a locally optimal blocking scheme s is learned from S over the training set, according to a specified PC threshold rather than a specific error rate (input and line 11). Then the scheme extension strategy is applied to determine whether a conjunction or disjunction form of this

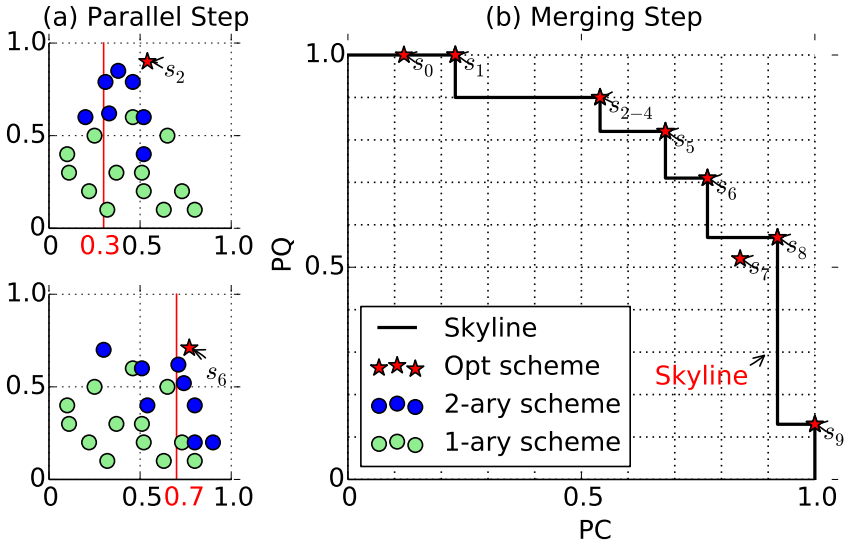

Figure 5.2: An illustration of the Naive Skyline Learning (Naive-Sky) algorithm. The optimal blocking schemes are learned in parallel as shown in (a), and the scheme skyline is depicted in (b).

optimal blocking scheme and other blocking schemes will be used based on Lemma 5.3.1 (lines 12-16). If a locally optimal blocking scheme is found, new blocking schemes are generated by extending s to a conjunction with each of the blocking schemes in S_{prev}, so that the PQ values of new blocking schemes may be further increased (lines 12 and 13). Otherwise a blocking scheme with the maximum PC value is selected and new schemes are generated using disjunctions, so that the PC values of new schemes can be further increased (lines 14 to 16).

The *Naive Skyline Learning (Naive-Sky)* algorithm uses the ASL+ procedure to learn optimal blocking schemes under different PC thresholds. A high-level description for this algorithm is described in Algorithm 3. The input that a user needs to specify is the label budget, the maximal ary of schemes in scheme skylines, and the size Δ of threshold interval, i.e., the difference of two consecutive thresholds. For example, if Δ is set to be 0.1, there will be in total 10 thresholds used, i.e., 0.1, 0.2, ..., 1.0, to learn at most 10 optimal blocking schemes. It is possible that the same blocking scheme is learned under two different thresholds. As shown in Algorithm 3, after initialization (line 1), this approach consists of two steps:

- *Parallel step*: Optimal blocking schemes are learned in parallel under different thresholds using the ASL+ procedure (lines 2-5).
- *Merging step*: All optimal blocking schemes are merged and dominance between them is checked, leading to generating a scheme skyline (line 6).

Note that, in our Naive-Sky algorithm, the sample size k is uniformly decided by the total label budget $budget(\zeta)$, the threshold interval Δ, and the number of predicates $|P|$. Fig. 5.2(a) illustrates how two optimal blocking schemes are learned in parallel, where their thresholds are set to 0.3 and 0.7, respectively. In Fig. 5.2(b), optimal schemes that are learned under the

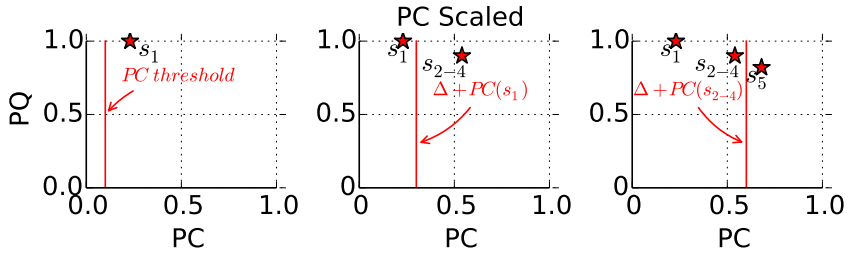

Figure 5.3: An illustration of the Adaptive Skyline Learning (Adap-Sky) algorithm. This algorithm can choose the PC threshold adaptively, based on the same example as in Fig. 5.2.

thresholds 0.1, 0.2, ..., 1.0 are merged, and a scheme skyline is generated. In Fig. 5.2, s_7 is dominated by the skyline, and $s_2 - s_4$ are the same scheme shown as s_{2-4}.

Algorithm 3: Naive Skyline Learning (Naive-Sky)

Input: Dataset: R
Human oracle ζ with $budget(\zeta)$
Set of blocking predicates P
Size of threshold interval Δ
Sample size k
Output: Scheme skyline S^*

1 $\epsilon = \Delta$, $S^* = \varnothing$, $S = \varnothing$
2 **while** $\epsilon \leq 1$ **do**
// Parallel step
3 $s = \text{ASL}(R, \epsilon, \zeta, P, k)$
4 $S = S \cup \{s\}$
5 $\epsilon += \Delta$
6 $S^* = \text{SKYLINE\_SCHEMES}(S)$ // Merging step
7 **Return** S^*

5.3.3 Adaptive Skyline Learning

Although the Naive-Sky algorithm offers the advantage of learning blocking schemes in parallel, it still has some limitations. For example, how to choose an appropriate threshold interval? If Δ is set too high, blocking schemes in a scheme skyline will be sparse and thus lack of accuracy. On the contrary, if Δ is set too low, users may have a large number of iterations being involved in learning blocking schemes under different thresholds, and some of them are redundant. This leads to unnecessary computational costs. A question arising naturally is whether the threshold interval Δ can be adjusted adaptively to improve the efficiency of algorithms but without sacrificing the accuracy of learned scheme skylines.

In our experiments with the Naive-Sky algorithm, we have noticed that the PC threshold specified by a user may not be (or even far from) the actual PC value of an optimal blocking

scheme being learned.

Example 5.3.1. *Let us consider Fig. 5.2 again. In Fig. 5.2(a), when the threshold is set to 0.3, the actual PC value of the learned optimal blocking scheme is 0.53. Moreover, when the threshold is set to 0.4 and 0.5, e.g.* $\Delta = 0.1$*, the learned optimal blocking schemes remain unchanged, i.e., corresponding to the same point* s_{2-4} *shown in Fig. 5.2(b).*

Based on this observation, we propose another algorithm for learning scheme skylines, called *Adaptive Skyline Learning (Adap-Sky)*, which can adaptively learn a scheme skyline even for a small Δ. The following lemma formulates the property behind this observation.

Lemma 5.3.2. *Let* s_ϵ *and* $s_{\epsilon'}$ *be two optimal schemes learned using ASL+ under the PC thresholds* ϵ *and* ϵ'*, respectively, and under the same label budget. The following property holds:*

$$s_{\epsilon'} = s_\epsilon, \quad \forall \epsilon' \in [\epsilon, PC(s_\epsilon)] \tag{5.4}$$

where ϵ' *is a threshold from the range* $[\epsilon, PC(s_\epsilon)]$.

Proof. Given a threshold ϵ and a set of blocking schemes S ($\forall s \in S$, $PC(s) > \epsilon$), we have $s_\epsilon \in S$. Given any threshold ϵ', s.t. $\epsilon < \epsilon' \leq PC(s_\epsilon)$, and a set of blocking schemes S' ($\forall s' \in S'$, $PC(s') > \epsilon'$), we thus have $s_\epsilon \in S'$ and $s_{\epsilon'} \in S'$. Based on Definition 7, the optimal blocking scheme is unique. This means $s_\epsilon = s_{\epsilon'}$. □

A high-level description of the Adap-Sky algorithm is presented in Algorithm 4. By Lemma 5.3.2, the Adap-Sky algorithm can adaptively select the next PC threshold based on the PC value of the current optimal blocking scheme (line 5).

Example 5.3.2. *Consider an illustration of the Adap-Sky algorithm shown in Fig. 5.3. In the leftmost plot, if* $\Delta = 0.1$*, the threshold* $\epsilon = 0.1$ *at the beginning, and the optimal blocking scheme is marked as* s_1*. In the middle plot, the new threshold is set to be* $PC(s_1) + \Delta$*, and the optimal blocking scheme is marked as* s_{2-4}*. This is different from Algorithm 3 which would take* $\epsilon = 0.1 + \Delta$ *as the new threshold. In the rightmost plot, which indicates the third iteration of the algorithm, the new threshold is* $PC(s_{2-4}) + \Delta$*.*

5.3.4 Progressive Skyline Learning

The Adap-Sky algorithm has reduced the number of iterations needed in Naive-Sky. However, it still requires to construct a training set in each iteration, and these training sets are constructed independently. We notice that some samples used in one ASL+ procedure may also be used in other ASL+ procedures in different iterations. For example, if a sample is selected twice when PC is set to 0.3 and 0.7, its label is counted twice. This gives rise to a new question: can we further eliminate duplicate samples occurring in the iterative process of constructing a training set so as to reduce the label cost?

To address the above question, we propose a *Progressive Skyline Learning* (Pro-Sky) algorithm. This algorithm can learn a scheme skyline without iteratively applying the ASL+ procedure. Instead, it learns a n-ary scheme skyline progressively, i.e., it starts by learning a 1-ary scheme skyline, then by learning a 2-ary scheme skyline, and finally by learning a n-ary

Algorithm 4: Adaptive Skyline Learning (Adap-Sky)

Input: Dataset: R
Human oracle ζ with $budget(\zeta)$
Set of blocking predicates P
Size of threshold interval Δ
Sample size k

Output: Scheme skyline S^*

$\epsilon = \Delta$, $S^* = \varnothing$, $S = \varnothing$
while $\epsilon \leq 1$ **do**
  $s = \text{ASL}(R, \epsilon, \zeta, P, k)$
  $S = S \cup \{s\}$
  $\epsilon = \text{PC}(s) + \Delta$
$S^* = \text{SKYLINE\_SCHEMES}(S)$
Return S^*

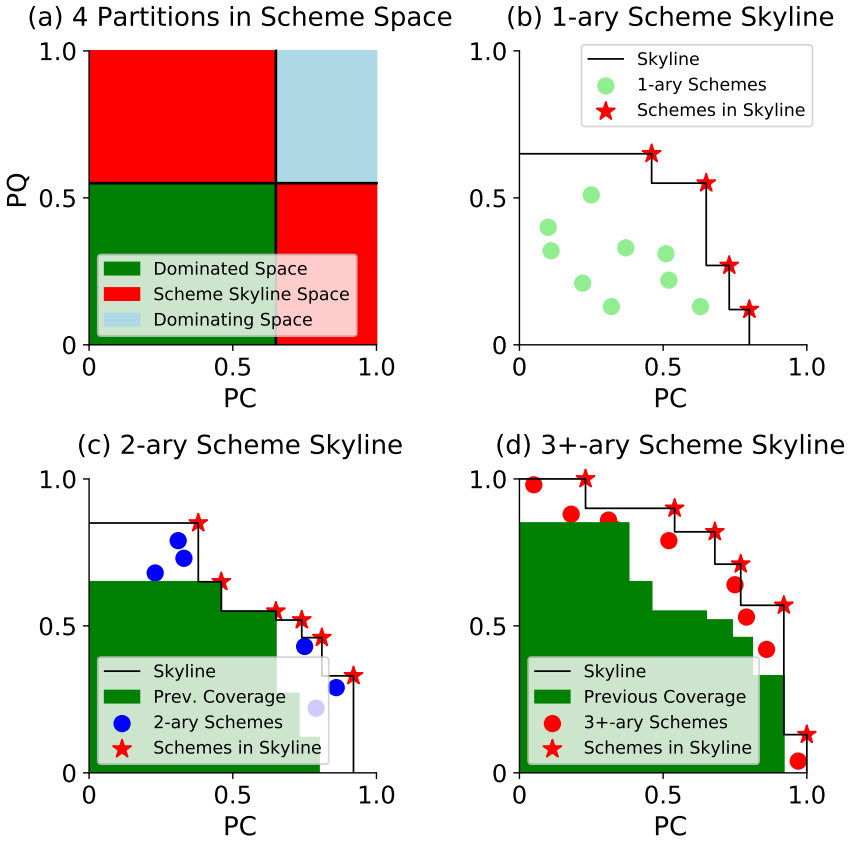

Figure 5.4: An illustration of the Progressive Skyline Learning (Pro-Sky) algorithm: (a) shows three different spaces w.r.t. a blocking scheme at the crossing; (b) - (d) illustrate how our Pro-Sky algorithm may learn a scheme skyline progressively.

scheme skyline, as illustrated in Fig 5.4. A user does not need to pre-define a threshold interval Δ.

As shown in Fig. 5.4(a), given a blocking scheme at the crossing, we can partition a scheme space into four parts. Any blocking schemes mapped in the green area, called *Dominated Space*, are dominated by the given blocking scheme. The blocking schemes mapped in the red area, called *Scheme Skyline Space*, may possibly appear in a skyline. If a blocking scheme is mapped in the blue area, called *Dominating Space*, it will replace the given blocking scheme in the skyline. Therefore, our objective for progressive skyline learning is: given a scheme skyline S^*, we try to extend each $s \in S^*$ by discarding schemes in its dominated space, verifying schemes in its scheme skyline space, and finding schemes in its dominating space.

A high-level description for the Pro-Sky algorithm is described in Algorithm 5. The sampling steps (lines 1-11) remain the same as in the ASL+ procedure. However, when extending a scheme, this algorithm takes the property illustrated in Fig. 5.4(a) into consideration. Each scheme in a skyline is extended with all the predicates that are not contained by the scheme (lines 12-13). By conducting both conjunctions and disjunctions, we obtain both PC and PQ values of the new schemes, and select ones with incremental PC or PQ values (line 14). That is to say, if a new scheme falls into the red area as shown in Fig. 5.4(a), we would add this scheme into the candidate set. If it appears in the blue area, we would replace the previous one by this new scheme. If it appears in the green area, we would discard it.

Algorithm 5: Progressive Skyline Learning (Pro-Sky)

Input: Dataset: R
Human oracle ζ with $budget(\zeta)$
Set of blocking predicates P
Ary of schemes l
Output: Scheme skyline S^*

1 $S = P, n = 0, T = \varnothing, X = \varnothing, k = \frac{budget(\zeta)}{2l \times |P|}$
2 $X = X \cup \text{RANDOM\_SAMPLE}(R)$
3 **while** $n < budget(\zeta)$ **do**
4 **for** *each* $s_i \in S$ **do** // Begin sampling
5 **if** $\beta(s_i, X) \leq 0$ **then**
6 $X = X \cup \text{SIMILAR\_SAMPLE}(R, s_i, k)$
7 **else**
8 $X = X \cup \text{DISSIMILAR\_SAMPLE}(R, s_i, k)$
9 $n = |X|$ // End sampling
10 $T = T \cup \{(x_i, \zeta(x_i)) | x_i \in X\}$ // Add samples
11 $S^* = \text{SCHEME\_SKYLINE}(S); S = \varnothing$
12 **for** $s_j \in S^*$ **do**
13 **for** $p_i \in P$ **do**
14 $\text{SELECT\_SCHEMES}(p_i, s_j, S)$
15 **Return** S^*

5.4 Theoretical Analysis

In this section, we discuss the search complexity and the time complexity for learning a scheme skyline.

Search complexity. To analyze the search complexity of the algorithms Naive-Sky and Adap-Sky, we first analyze the complexity of ASL, which is $O(n^2)$ as has been discussed in Section 4.4. Hence, for a given Δ, the search complexity of Naive-Sky is $O(\frac{n^2}{\Delta})$, and in the worst case, the search complexity of Adap-Sky is the same as $O(\frac{n^2}{\Delta})$. Nonetheless, the search complexity of the algorithm Pro-Sky is different. If we use $|Opt_i|$ to describe the number of i-ary schemes being selected, then the search complexity of Pro-Sky will be $\sum_{i=1}^{n} |Opt_i| \times 2|P|$, where 2 indicates both conjunction and disjunction of schemes. In the worst case that all schemes are in the skyline, this is the same as *Dedekind Number*. However, if we further introduce a Δ in this algorithm, i.e., there should be at least a distance of Δ between two schemes in a skyline, the search complexity of Pro-Sky will be reduced to $\frac{n^2}{\Delta}$ ($\frac{1}{\Delta} \leq n$) in the worst case.

Time complexity. Since for all the three algorithms in skyblocking, we adopt active sampling strategy proposed in Section 4.3.1, the time complexity for sampling is $O(|R| + k)$ for a blocking predicate.

In our experiments (will be further discussed in Section 5.5), we notice that blocking predicates are sufficient to conduct blocking efficiently and effectively for most datasets. Only for Amazon-GoogleProducts, similarity-based blocking predicates may bring some benefits by trading off efficiency to certain degree. This also suggests that, by relaxing the definition of blocking predicate, our framework can be generalized to learn scheme skylines not only for blocking, but also for entity resolution itself, over various datasets.

5.5 Experiments

We have conducted experiments to empirically verify our skyline learning algorithms, aiming to answer the following questions:

(1) Given a limited label budget, how does our skyline learning approach perform w.r.t. the running time under different threshold intervals and different PC thresholds?

(2) How does our skyline learning approach perform compared with the baselines w.r.t. PC, PQ, RR and FM?

(3) How does our skyline learning approach perform in reducing label budgets while still achieving the same level of quality for blocking as the baseline methods?

5.5.1 Experimental Setup

In the following, we present the datasets, blocking predicates, baseline approaches, and measures used in our experiments.

Datasets. We have used five datasets in our experiments: *Cora*, *DBLP-Scholar*, *DBLP-ACM*, *NCVoter* and *Amazon-GoogleProducts* [95]. The characteristics of the first four datasets are

introduced in Table 2.2, and the referring attributes are summarized in Table 2.3. Additionally, *Amazon-GoogleProducts* dataset contains product entities from the online retailers Amazon.com and the product search service of Google accessible through the Google Base Data API. It contains 1,363 and 3,226 records, respectively with the number of true matches 1,291, with a class imbalance ratio of 1 : 3,405. We use 20 blocking predicates for this dataset corresponding to 4 attributes.

Blocking predicates. We have used five blocking functions in our experiments. four blocking functions are described in Section 4.5.1 to obtain 16 or 72 different blocking predicates for *Cora*, *DBLP-Scholar*, *DBLP-ACM* and *NCVoter*. Additionally, we use the blocking function *Top-similarity* for the dataset *Amazon-GoogleProducts* to obtain a total number of 20 blocking predicates due to the following two reasons: (1) it has a higher time complexity since it needs to compare the similarity of two records rather than compare the string values of two records, and (2) it has little effects on the performance of the other datasets. This blocking function takes two strings as input and returns 1 if one string is most similar to the other; otherwise, return 0.

Baselines. We have used the following approaches as the baselines: (1) *Fisher* [84], which is the state-of-the-art unsupervised scheme learning approach proposed by Kejriwal and Miranker. (2) *TBlo* [47], which is a blocking approach based on expert-selected attributes. In the survey [27], this approach has a better performance than the other approaches in terms of the F-measure results. (3) *RSL (Random Scheme Learning)*, which is an algorithm that has the same structure of the ASL+ procedure except that random sampling is used to build a training set, instead of active sampling. In each experiment, we have run the RSL ten times. We present the average results of the blocking schemes it has learned. As reported in [84], Fisher has shown a better performance compared with the supervised blocking scheme learning approach [11]. We thus do not consider this supervised learning approach as a baseline.

Measures. The measures we use in the experiments for this chapter include *Pairs Completeness (PC)*, *Pairs Quality (PQ)*, *F-measure* and *Constraint Satisfaction (CS)*, which are introduced in Section 2.2.2 and Section 4.5.1 in details.

5.5.2 Results and Discussion

In this section, we will discuss the experimental results of our skyline algorithms, and compare the performance of our algorithms with the baseline approaches.

5.5.2.1 Label Efficiency

We have first conducted experiments to evaluate the label efficiency of our algorithms.

Label costs. The label costs of our proposed algorithms, namely *Naive-Sky* for Algorithm 3, *Adap-Sky* for Algorithm 4 and *Pro-Sky* for Algorithm 5, are shown in Table 5.1, where the label costs of these algorithms for learning scheme skylines of five datasets with $CS = 90\%$ are recorded. We consider two variants of *Naive-Sky*: sequential *Naive-Sky* and parallel *Naive-Sky*.

- Sequential *Naive-Sky* iteratively computes optimal blocking schemes with the PC threshold being increased by Δ in each iteration.
- Parallel *Naive-Sky* computes a number of optimal blocking schemes in parallel, where a range of thresholds from 0 to 1 with interval Δ are assigned to these parallel computations (one threshold for each computation).

We set $\Delta = 0.1$ and $\Delta - 0.05$ for all datasets. For example, if $\Delta = 0.1$, sequential *Naive-Sky* iterates 10 times with the PC threshold being increased by 0.1 at each time, and parallel *Naive-Sky* takes 10 parallel computations, each computation deals with the ASL+ procedure once with a PC threshold ranging from 0.1 to 1. Furthermore, for *Naive-Sky* and *Adap-Sky*, the label budget begins with 50, and increases by 50 for each run of ASL. The total label cost is accumulated when the PC threshold increases. For *Pro-Sky*, the label budget begins with 500, and increases by 500. This is to ensure a fair comparison with *Naive-Sky* and *Adap-Sky* because, when Δ is set to 0.1, *Naive-Sky* needs exactly ten iterations to reach 1 from 0.1, and *Adap-Sky* needs at most ten iterations to reach 1 from 0.1. Thus, the total label consumption of these two algorithms is $50 \times 10 = 500$ in the worst case. Since *Pro-Sky* does not involve iterations, we set its sample size to 500. The label cost is recorded when its CS value reaches 90% in ten consecutive runs. Table 5.1 shows that *Adap-Sky* uses less labels than *Naive-Sky*; nonetheless, *Pro-Sky* can reduce the label cost from one third to a half of the label cost of *Adap-Sky*.

Observation 1. *Our Pro-Sky algorithm takes the smallest number of labels compared with Naive-Sky and Adap-Sky to learn the scheme skyline under specific threshold intervals.*

Table 5.1: Comparison on the label costs and run times (in seconds) of three skyline algorithms over five datasets. Δ refers to the threshold interval and *RT* refers to the run time in seconds.

	Cora			DBLP-Scholar			DBLP-ACM			NCVoter			Amazon-GoogleProducts		
	Budget	Δ	RT	Budget	Δ	RT	Budget	Δ	RT	Budget	Δ	RT	Budget	Δ	RT[3]
Naive-Sky (Sequential)	6000	0.1	18.74	5000	0.1	96.71	3000	0.1	76.95	3500	0.1	225.08	5000	0.1	98.3
		0.05	21.55		0.05	133.25		0.05	90.8		0.05	267.85		0.05	120.47
Naive-Sky (Parallel)	6000	0.1	12.19	5000	0.1	35.51	3000	0.1	50.6	3500	0.1	83.92	5000	0.1	72.17
		0.05	18.33		0.05	40.5		0.05	73.51		0.05	110.81		0.05	93.68
Adap-Sky	4200	0.1	12.48	3500	0.1	56.28	1250	0.1	33.23	1400	0.1	118.4	1600	0.1	48.93
		0.05	14.68		0.05	60.73		0.05	34.55		0.05	122.16		0.05	60.36
Pro-Sky	2500	0.1	5.6	2000	0.1	32.56	1000	0.1	11.38	1000	0.1	63.09	1000	0.1	17.33
		0.05	6.27		0.05	38.09		0.05	13.28		0.05	70.56		0.05	20.1

Table 5.2: Comparison on the label costs of ASL+ and RSL with CS = 90%.

PC Threshold	Cora	DBLP-Scholar	DBLP-ACM	NCVoter	Amazon-GoogleProducts
0.2	600	500	300	300	400
0.4	**400**	350	200	350	400
0.6	450	**250**	**150**	250	250
0.8	550	300	200	**200**	**200**
0.9	500	**250**	300	250	300
RSL	7,900	10,000+	2,200	10,000+	5,000

Label efficiency analysis. We have also analyzed the factors that may affect the label costs. First, as explained before, both extremely high and low PC thresholds may require more labels than in other cases. Second, datasets of a smaller size (i.e., containing a smaller number of record pairs) often need less labels. Furthermore, datasets with more attributes have a larger number of blocking predicates and thus need a higher label budget in order to be able to consider all blocking predicates in sampling. The quality of a dataset may also affect the label costs. Generally, the cleaner a dataset is, the less labels it requires. For example, although Cora has the smallest number of attributes and records, it still needs the largest number of labels because it contains many fuzzy values (e.g. mis-spelling names, first name and last name being swapped and etc.). On the contrary, NCVoter needs a lower label budget although it has more attributes and records than several other datasets. The cleanness of a dataset also affects the distribution of label costs under different thresholds. For example, with a PC threshold $\epsilon = 0.2$, we need only 200 labels for NCVoter but 550 labels for Cora to generate blocking schemes with CS = 90%.

Observation 2. *Our skyline learning approach using active sampling strategy can use much less labels to achieve a target constraint satisfaction compared with random sampling strategy.*

5.5.2.2 Time Efficiency

Table 5.1 shows Run Time (RT) of our three algorithms over five datasets, in relation to different label budgets. Two threshold intervals are used, i.e., $\Delta = 0.05$ and $\Delta = 0.1$. Generally speaking, the RT values depend on two factors: (1) the size of a dataset and (2) the number of samples. With the increase of the label budget and the number of records in a dataset, the RT value increases. We can also see that, for these three algorithms, the threshold interval Δ does affect the run time, but not significantly.

We also present the RT values of *Naive-Sky* in two different settings: running sequentially and running in parallel. The parallel method can reduce the run time considerably, varying from 30% to 70%. Especially, for the DBLP-Scholar and NCVoter datasets, *Naive-Sky* running in parallel can even outperform *Adap-Sky*.

Observation 3. *Our skyline learning approach is quite efficient for all the datasets under a limited number of label budgets.*

[3] A pre-calculated similarity matrix is used for searching most similar records; hence, RT does not consider the time consumption of similarity computation.

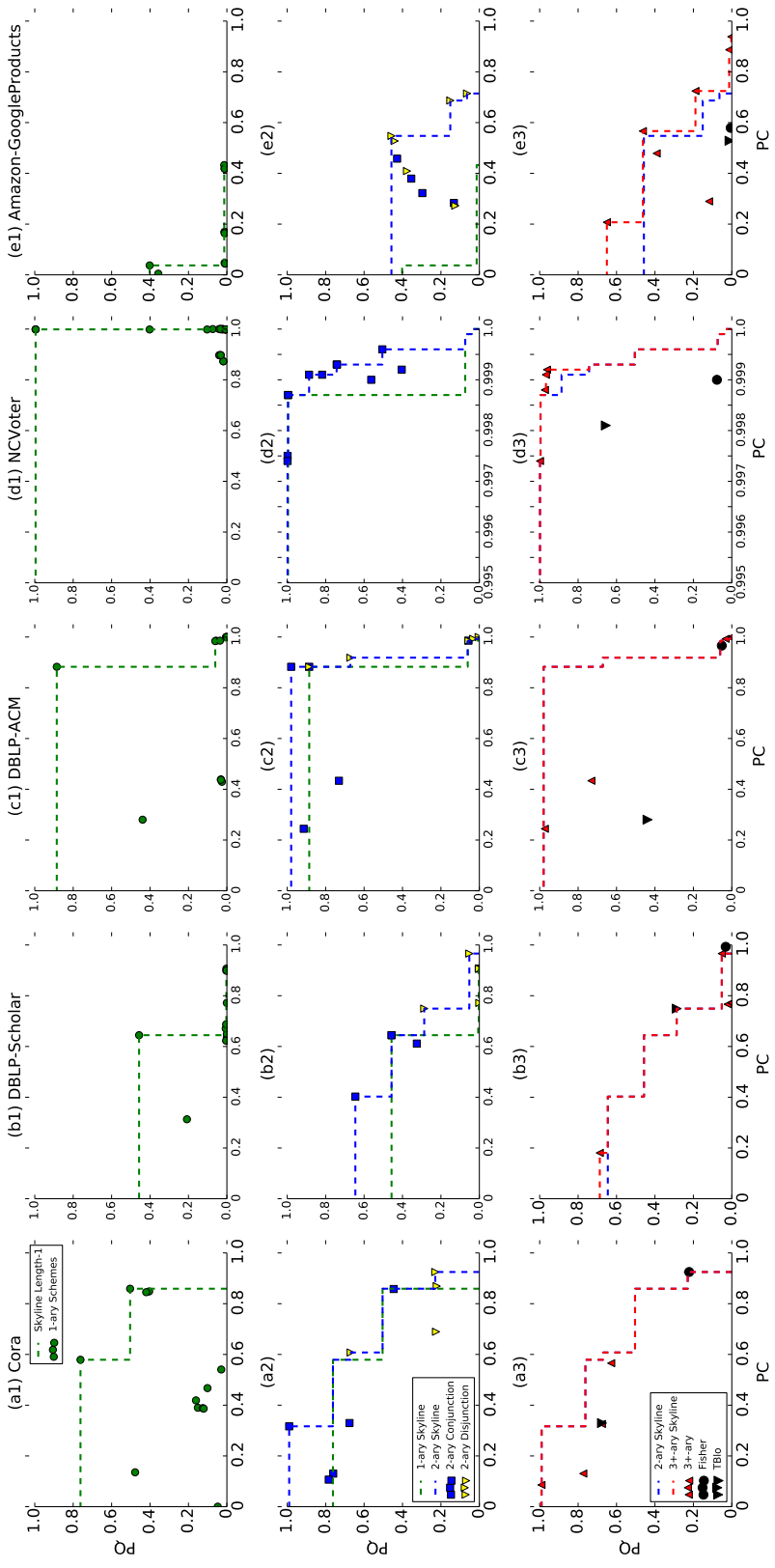

Figure 5.5: An illustration of the progressive process for learning scheme skylines by *Pro-Sky* over five datasets. The three rows from top to bottom show the results of 1-ary, 2-ary (in both conjunction and disjunction) and 3-ary blocking schemes, respectively.

5.5.2.3 Blocking Quality

Now we discuss how the blocking quality of our skyline algorithms may be affected by the choice of n-ary blocking schemes, as well as PC thresholds. We also compare the quality of blocks generated by our skyline algorithms with the quality of blocks generated by the baseline approaches.

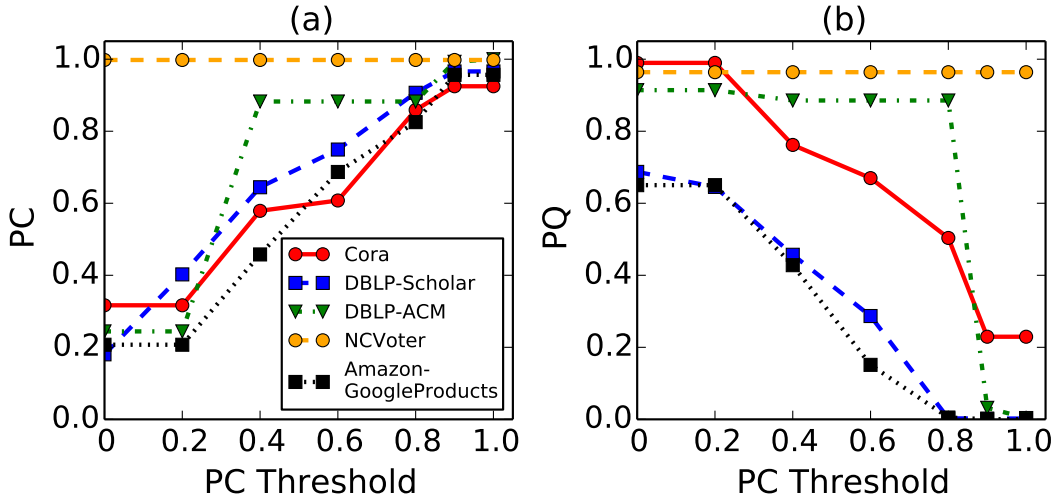

Figure 5.6: Comparison on blocking quality using *Pro-Sky* under different PC thresholds over five datasets: (a) PC and (b) PQ.

Under different number of aries. The experimental results of the *Pro-Sky* algorithm under different number of aries are presented in Fig. 5.5. We have also tested on *Naive-Sky* and *Adap-Sky* with $\Delta = 0.05$ and $\Delta = 0.10$. However, because the blocking schemes in the scheme skylines learned by these algorithms are the subsets of the scheme skyline generated by *Pro-Sky* in which no Δ is defined, we thus omit the results for *Naive-Sky* and *Adap-Sky*. Fig. 5.5 illustrates how *Pro-Sky* can progressively generate the scheme skylines using blocking schemes from 1-ary to 3+-ary over five datasets. We do not present the further process of 3+-ary, as the scheme skylines have already been generated within 3-ary and they would remain the same. In the plots (d2) and (d3) of Fig. 5.5, we present the scheme skylines with $PC \in [0.995, 1]$ because most of the schemes on the skyline for NCVoter dataset are located in this range.

The *Pro-Sky* algorithm can detect a number of 1-ary schemes and generate the scheme skylines as shown in the plots (a1), (b1), (c1) and (d1) of Fig. 5.5, although some of the learned blocking schemes are not on the skylines. When considering 2-ary blocking schemes in both conjunction and disjunction, we notice that disjunction schemes have higher PC values but lower PQ values compared with conjunction ones. Some 2-ary blocking schemes have better performance than 1-ary blocking schemes in the scheme skylines but some are not. Hence, a scheme skyline can be updated with some segments remaining the same, e.g. PC values from 0.4 to 0.63 in the plot (b2). The third row of the figure shows 3+-ary blocking schemes. The scheme skylines (red dashed lines) do not change much compared with the 2-ary scheme

skylines (blue dashed lines). We also show the results of the baseline approaches, where Fisher normally generates the blocking schemes with high PC values, but TBlo schemes are not in a skyline in the most cases.

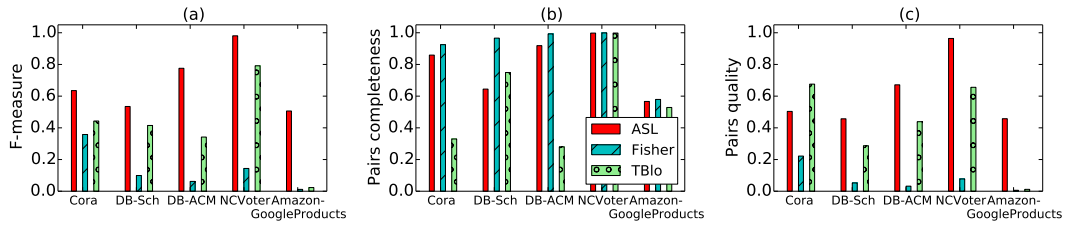

Figure 5.7: Comparison on blocking quality by using different blocking approaches over five datasets and using the measures: (a) FM, (b) PC, and (c) PQ.

Observation 4. *With the increasing of number of aries, the scheme skyline becomes higher, which means the schemes under smaller numbers of aries are dominated by the ones under larger numbers. That is to say, we can find blocking schemes with better performance with the increment of aries.*

Under different PC thresholds. To make a detailed comparison with the baseline approaches, we have conducted experiments based on the ASL+ procedure under different PC thresholds. We have monitored the blocking schemes learned by the ASL+ procedure as well as their PC and PQ values under different PC thresholds $\epsilon \in \{0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$. The results are presented in Fig. 5.6. We can see that the PC and PQ values may vary because the learned blocking schemes are different when the PC thresholds increase. The label budget is ensured to be sufficient in these cases.

With the increment of the PC thresholds, the blocking schemes we learn generate lower PC values and higher PQ values. However, there are still some overlapping points. This indicates that under certain thresholds, the *Pro-Sky* algorithm learns the same blocking scheme. On the other hand, this also provides further evidence showing the efficiency of *Adap-Sky* compared with *Naive-Sky*. For example, the performance of *Pro-Sky* for NCVoter dataset is consistent with both high PC and PQ values, whatever the PC threshold is. On the contrary, for other datasets such as Cora, a lower PC threshold allows the *Pro-Sky* algorithm to seek for blocking schemes that can generate higher PQ values, but the PC values decrease. We also notice that, for DBLP-ACM and NCVoter datasets, blocking schemes with both high PC and PQ values are learned with a low threshold (e.g. $\epsilon = 0.4$), but for DBLP-Scholar, no blocking schemes can be learned with high PQ values (i.e., higher than 0.6). In the figure, the blocking schemes with the PC threshold 1.0 are normally hard to learn, hence we present the maximum PC values of the blocking schemes learned by our algorithms.

Observation 5. *With the increasing of PC values as threshold, the PC value of the learned blocking scheme increases, while the PQ value of the learned blocking scheme decreases.*

Compared with baselines. Existing work largely focuses on learning a single blocking scheme, while our algorithms aim to learn a scheme skyline, which is a set of blocking

schemes. Hence we first conduct experiments to show that, in a 2-dimension space of PC and PQ, the blocking schemes learned by the baselines *TBlo* and *Fisher* are dominated by or contained in our scheme skylines over five datasets. The experimental results are shown in the plots (a3), (b3), (c3), and (d3) of Fig. 5.5.

Table 5.3: **Comparison on blocking quality.**

	PC		PQ	
	TBlo	ASL+	TBlo	ASL+
Cora	0.3296	0.3167	0.6758	0.9898
DBLP-Scholar	0.7492	0.7492	0.2869	0.2869
DBLP-ACM	0.2801	0.8826	0.4387	0.8854
NCVoter	0.9981	0.9979	0.6558	0.9640
Amazon-GoogleProducts	0.5285	0.5665	0.0118	0.4627

	PC		PQ	
	Fisher	ASL+	Fisher	ASL+
Cora	0.9249	0.9249	0.2219	0.2219
DBLP-Scholar	0.9928	0.9928	0.0320	0.0320
DBLP-ACM	0.9661	0.9686	0.0522	0.6714
NCVoter	0.9990	0.9990	0.0774	0.0774
Amazon-GoogleProducts	0.5792	0.7244	0.0059	0.1892

To make a fair point-to-point comparison with the baseline approaches, we have further conducted experiments which take the baseline PC values as the PC thresholds for the ASL+ procedure. Table 5.3 presents the PC and PQ values in the experimental results. Here, we set the PC threshold as the PC values used in the baseline approach. Compared with *TBlo*, our *Pro-Sky* algorithm can generate blocks with much higher PQ values (i.e., from 50% to 101%) while remaining similar PC values, except in *DBLP-Scholar* where the results are the same. Compared with *Fisher*, in the most cases, the results are the same, except in *DBLP-Scholar*, the results generated by our *Pro-Sky* algorithm have a 12 times higher PQ value. In general, our *Pro-Sky* algorithm can generate results as good as or better than the baseline approaches under the same thresholds and with a sufficient label budget.

We also present blocking schemes with the highest f-measure values learned by our algorithms and the baselines, and compare their FM, PC and PQ values. The FM results are shown in Fig. 5.7(a) in which our *Pro-Sky* algorithm outperforms all the baseline approaches over all the datasets. Since all the approaches yield high RR values over five datasets, the RR values are not presented in the figure. In Fig. 5.7(b), the PC values of our *Pro-Sky* algorithm are not the highest ones over the five datasets, but they are not much lower than the highest ones (i.e., within 10% lower except in DBLP-Scholar). Moreover, our *Pro-Sky* algorithm can generate higher PQ values than all the other approaches, from 15 percents higher in NCVoter (0.9956 vs 0.8655) to 20 times higher in DBLP-ACM (0.6714 vs 0.0320), as shown in Fig. 5.7(c).

Observation 6. *Our skyline learning approach achieves better blocking quality compared with the baselines under a 3+-ary blocking schemes.*

5.6 Summary

In this chapter, we have proposed a scheme skyline learning framework called skyblocking, which integrates skyline query techniques and active learning techniques into learning a set of optimal blocking schemes under different constraints and a limited label budget. We have tackled the class imbalance problem by solving the balanced sampling problem. We have also proposed the scheme extension strategy to reduce the searching space and label costs. We have further developed three algorithms for efficiently learning scheme skylines. Our algorithms overcome the weaknesses of existing blocking scheme learning approaches in two aspects: (1) Previous supervised blocking scheme learning approaches require a large number of labels for learning a blocking scheme, which is an expensive task for entity resolution; (2) Existing unsupervised blocking scheme learning approaches generate training sets based on the similarity of record pairs, instead of true labels, thus the training quality can not be guaranteed.

Chapter 6

Learning-To-Sample for Entity Resolution

6.1 Introduction

In entity resolution, a machine learning-based classifier is normally used to categorize entity resolution samples into matches and non-matches. During this process, sampling is a fundamental technique for acquiring training data. However, obtaining large amounts of manually labeled samples is often expensive or simply infeasible in practice. To alleviate this challenge, active learning has been extensively studied in the past decades [133], which aims to select fewer labeled samples to train a machine learning model as effectively as possible, achieving similar or even better accuracy. At its core, active learning seeks for the most representative or informative samples to be labeled for training by leveraging observations from previously labeled samples, such as uncertainty samples and diversity samples [40; 106; 38].

As in the literature, one limitation of most of the existing approaches is that the sampling strategies are pre-defined by humans [69]. For example, selecting samples which lie closest to the SVM's dividing hyperplane for SVM based active learning [132; 45]; Selecting samples by ranking such as estimated entropy [68]. These strategies are inflexible and need to be individually defined for different tasks [93]. Meta-learning algorithms for active learning are emerging as a promising paradigm for learning the "best" active learning strategy. However, current learning-based active learning approaches still require sufficient training data so as to generalize meta-learning models for active learning [69; 93]. This is contrary to the nature of entity resolution and active learning which typically starts with a small number of labeled samples. The unavailability of large amounts of labeled samples for training meta-learning models would inevitably lead to poor performance (e.g., instabilities and overfitting).

In this chapter, we aim to propose a learning-based active learning framework to enable a unified sampling process for selecting representative and informative samples under a limited number of labeled samples to solve entity resolution tasks. Different from the previous active learning approaches, we ground our work on the following observations: (1) Although uncertainty sampling is one of the widely used active learning techniques [98], this strategy alone tends to select samples that are similar to each other, i.e., samples being selected from a sample space often have similar features [157]. (2) Diversity sampling targets to select samples of different kinds (e.g., samples with different features), which is complementary to uncertainty

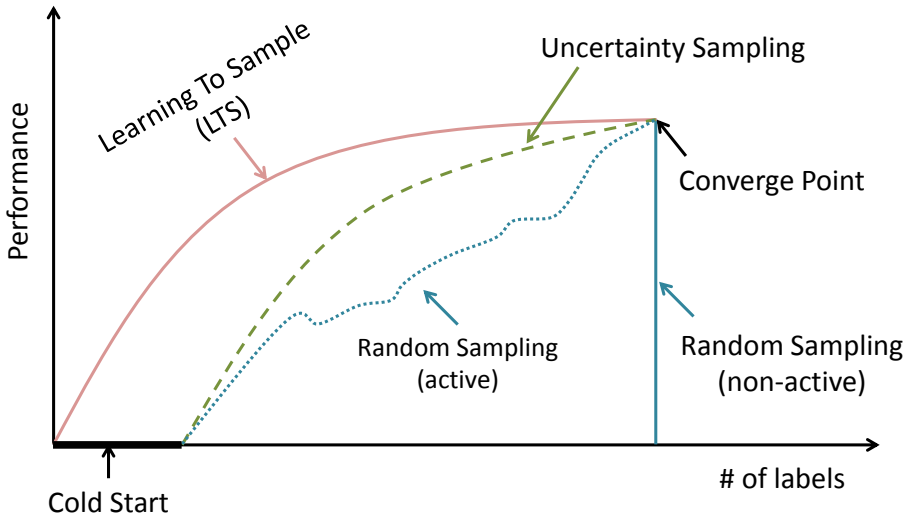

Figure 6.1: An illustration of Learning-To-Sample (LTS) for entity resolution in relation to uncertainty sampling and random sampling, where random sampling (active) indicates that random samples are gradually selected during the iterations of active learning, and random sampling (non-active) indicates that all samples are randomly selected in a one-off manner (i.e., no active learning).

sampling. Thus, the obstacle of uncertainty sampling can be circumvented by combining uncertainty sampling and diversity sampling into a unified sampling process. (3) To find the "best" way to integrate these two sampling strategies, meta-learning is a powerful tool, which can optimize this integration process by learning hints from the chosen machine learn models and datasets.

Based on the above observations, we design a novel learning-based active learning framework, called *Learning-To-Sample* (LTS). In a nutshell, the LTS framework consists of two key components: a sampling model G and a boosting model F, which can dynamically learn from each other in iterations for improving the performance of each other. As illustrated in Fig. 6.1, the goal of this LTS framework is to help machine learning-based entity resolution models achieve better performance with less training data by providing a learning-based active learning process. The design of the LTS framework, especially the sampling model, incorporates the uncertainty and diversity aspects of sampling into a unified process for optimization. This allows us to actively select samples based on the joint impacts of probabilities of being mis-classified by a boosting model and the distribution of samples in a sample space and circumvent the cold start problem at the same time [38; 93]. The experimental results show that our active learning approach significantly outperforms all the baselines when the label budget is limited, especially for those datasets with highly imbalanced classes. It also shows that our approach can effectively tackle the cold start problem.

The rest of this chapter is structured as follows. Section 6.2 introduces the formulation of our classification task for entity resolution. Section 6.3 introduces our Learning-To-Sample

framework, which includes a boosting model and a sampling model. Section 6.4 presents how the sampling model actively selects samples w.r.t. their uncertainty and diversity, and how some hyper-parameters are decided in the algorithm. Section 6.5 analyzes the reason that LST can alleviate the cold start problem. Then Section 6.6 discusses our experimental results. We conclude the chapter in Section 6.7.

6.2 Problem Formulation

Our task is to tackle the entity resolution classification problem with active learning as below.

Definition 8. *Let $X \subseteq \mathbb{R}^d$ contain $|X|$ samples and $budget(\zeta)$ be a budget on the total number of samples that can be labeled by a human oracle. A training set $T = \{(x_i, y_i)\}_{i=1}^{|T|}$, where $x_i \in X$ and $y_i \in \mathbb{R}$, consists of a set of samples from X and their labels from $\mathbb{R}$. The **Learning-To-Sample** problem is to actively select a set of samples forming a training set T, and train a model Λ that can predict the labels $\hat{y}$ for samples w.r.t.*

$$\begin{aligned} &\textbf{\textit{maximize}} \quad E(\Lambda) \\ &\textbf{\textit{subject to}} \quad |T| \leq budget(\zeta) \end{aligned} \tag{6.1}$$

where $E(\Lambda) = \sum_{x \in X \wedge \hat{y} = y} 1$

Intuitively, $E(\Lambda)$ refers to the total number of samples whose labels are correctly classified by Λ, and the size of training samples should be no more than the budget.

6.3 The Learning-To-Sample Framework

In this section, we present our learning-based active learning framework, called *Learning-To-Sample (LTS)*. As illustrated in Figure 6.2, the LTS framework has two key components: a boosting model F and a sampling model G. Accordingly, there are two learning processes that are closely coupled: (1) learning the boosting model F, and (2) learning the sampling model G. Specifically, a boosting model F aims to create a strong learner based on a set of weak learners. Thus, we have designed an incrementally built training set by adding a sequence of subsets, so that new functions can also be trained and added into the boosting model F for performance improving. Samples in these training sets are actively selected by the sampling model G which is dynamically learned from the performance of the boosting model F during its iterative training process. In the following, we discuss them in detail.

It is worth noting that, technically, the boosting model F can be replaced by any classification model and the regressors in the sampling model G can be replaced by any regression model. Thus, the LTS framework is indeed not restricted to specific machine learning models used for classification and regression.

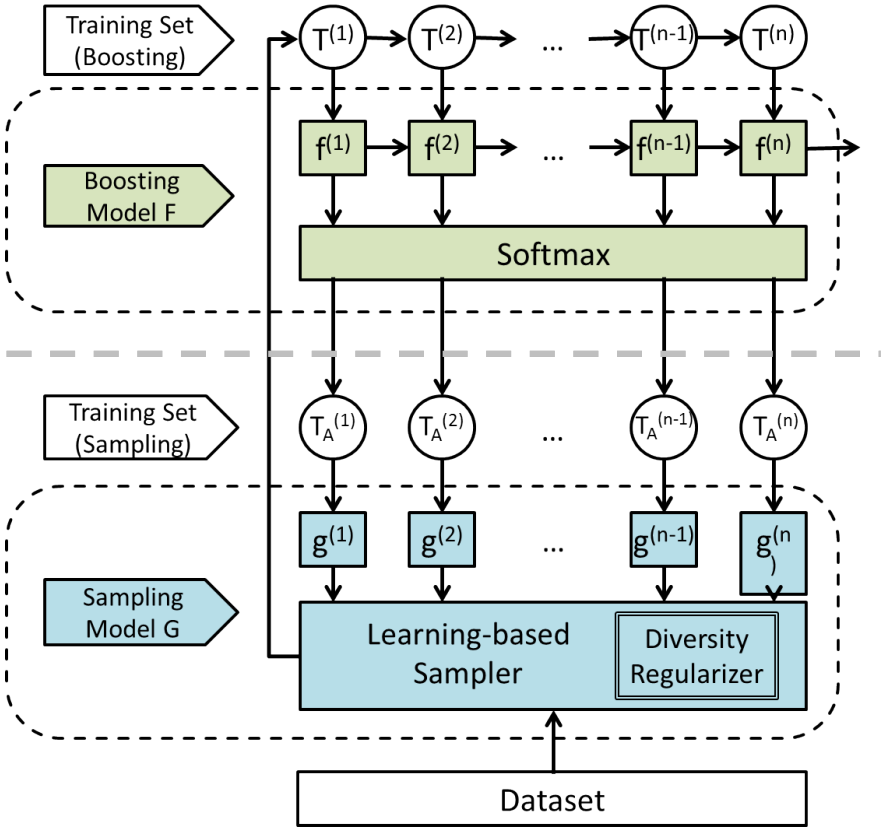

Figure 6.2: Overview of the LTS framework. The boosting model is highlighted in green and the sampling model is highlighted in blue.

6.3.1 Boosting Model

To actively select a set of training samples, we have built a training set T that is incrementally built as the boosting model interacts with the sampling model, i.e., a sequence of training subsets $\langle T^{(1)}, \ldots, T^{(n)} \rangle$ such that $T^{(1)} \subseteq T^{(2)} \subseteq \cdots \subseteq T^{(n)}$, $T^{(n)} = T$, and $|T^{(n)}| \leq budget(\zeta)$, where $T^{(t)}$ for $t \in [1, n]$ is a training subset being used for training the boosting model at the t-th iteration.

A boosting model F trains a sequence of functions $\langle f^{(1)}, \ldots, f^{(n)} \rangle$ in an additive manner, where $f^{(t)}$ for $t \in [1, n]$ is a function being added into F at the t-th iteration. More specifically, the individual results of the first t-1 functions are combined to predict the label of a sample at the $(t$-1$)$-th iteration such that:

$$\hat{y}_i^{(t-1)} = \sum_{k=1}^{t-1} f^{(k)}(x_i). \tag{6.2}$$

Then, the t-th function $f^{(t)}$ is trained on the actively selected training subset $T^{(t)}$ by minimizing

the following objective function:

$$\sum_{(x_i,y_i)\in T^{(t)}} \ell_1(\hat{y}_i^{(t-1)} + f^{(t)}(x_i), y_i) + \Omega_1(f^{(t)}) \tag{6.3}$$

where ℓ_1 is a differentiable loss function and $\Omega_1(f^{(t)})$ is the penalty for the complexity of $f^{(t)}$.

After the t-th function $f^{(t)}$ is learned, the boosting model F sends its feedback to the sampling model G via a softmax layer. This allows the sampling model G to leverage hints from the prediction results of $\langle f^{(1)}, \ldots, f^{(t)} \rangle$ and actively select the most informative samples as new samples for the next iteration, leading to $T^{(t+1)}$. We use the *Softmax* function [139] to obtain probabilities of being mis-classified for training samples. Specifically, in the t-th iteration, the softmax layer takes $\mathbf{l}^{(t)} = \langle \ell(\hat{y}_1^{(t)}, y_1), \ldots, \ell(\hat{y}_q^{(t)}, y_q) \rangle$ as input, where $q = |T^{(t)}|$ and each $\ell(\hat{y}_j^{(t)}, y_j)$ in $\mathbf{l}^{(t)}$ refers to the loss of a training sample x_j from $T^{(t)}$, then generates $\mathbf{z}^{(t)} = \langle z_1^{(t)}, \ldots, z_q^{(t)} \rangle$, i.e.,

$$z_i^{(t)} = \mathit{Softmax}(l_i^{(t)}), \tag{6.4}$$

where $\mathit{Softmax}(l_i^{(t)}) = e^{l_i^{(t)}} / \sum_{j=1}^{q} e^{l_j^{(t)}}$ and $l_i^{(t)} = \ell(\hat{y}_i^{(t)}, y_i)$.

6.3.2 Sampling Model

Let $X^{L(t)} = \{x_i \in X | (x_i, y_i) \in T^{(t)}\}$ be the set of labeled samples and $X^{U(t)} = X - X^{L(t)}$ be the set of unlabeled samples in the t-th iteration. A sampling model G aims to select a set $\Delta^{(t)}$ of the most informative samples from unlabeled samples at the t-th iteration such that $X^{L(t+1)} = X^{L(t)} \cup \Delta^{(t)}$ and $X^{U(t+1)} = X^{U(t)} - \Delta^{(t)}$. Consequently, $T^{(t+1)} = T^{(t)} \cup \{(x_i, y_i) | x_i \in \Delta^{(t)}\}$ is generated and sent to the boosting model F for training the function f^{t+1}.

The question arising here is: how to actively select a set $\Delta^{(t)}$ of the most informative samples at the t-th iteration? In the LTS framework, two kinds of samples are primarily targeted: (1) samples that are likely to be mis-classified by the boosting model; (2) samples that have diverse features in the sample space. They relate to the uncertainty and diversity aspects of sampling, respectively. Hence, at the t-th iteration, the sampling model G learns to select a set $\Delta^{(t)}$ of most informative samples by maximizing the following objective:

$$\begin{aligned} \textbf{maximize} \quad & \sum_{i=1}^{k} v_i g^{(t)}(x_i) + \alpha \times \Gamma(\mathbf{v}) \\ \textbf{subject to} \quad & ||\mathbf{v}||_1 = |\Delta^{(t)}| \end{aligned} \tag{6.5}$$

where $k = |X^{U(t)}|$, $\mathbf{v} = (v_1, ..., v_k)^T \in \{0,1\}^k$ is a binary vector, and each v_i is associated with a sample $x_i \in X^{U(t)}$. When $v_i = 1$, it indicates that x_i is selected as a sample, and conversely, $v_i = 0$ indicates that x_i is not selected. The term $g^{(t)}(x_i)$ indicates the uncertainty score of a sample x_i which is predicated by a regressor $g^{(t)}$, and the regularization term $\Gamma(\mathbf{v})$ controls the distribution of selected samples in order to ensure their diversity in the sample

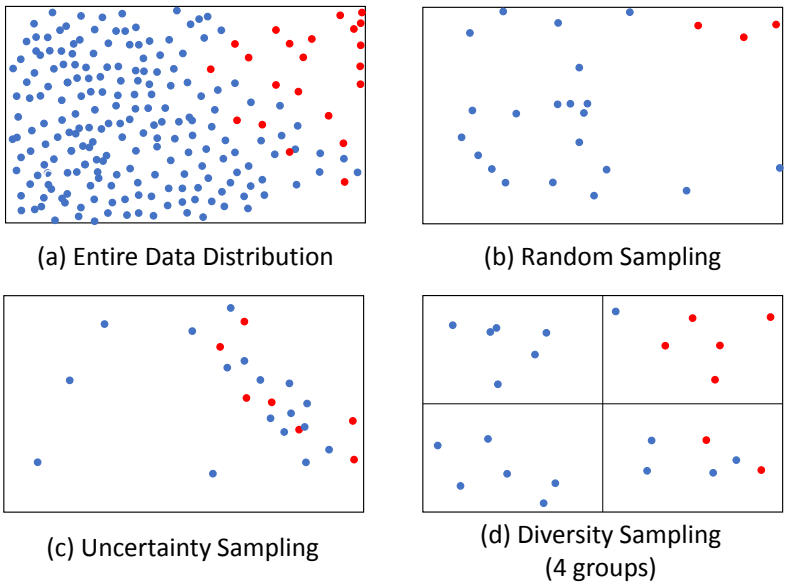

Figure 6.3: Comparison of different sampling strategies. 24 samples are selected in each sub-figure of (b), (c) and (d).

space. α is a parameter used for balancing the impacts of uncertainty and diversity on samples, i.e., $\alpha > 1$ indicates that diverse samples are preferred, while $\alpha < 1$ indicates that samples with high probabilities of being mis-classified are preferred. Further details for our sampling model will be discussed in the next section.

6.4 Sampling Strategies

In the following, we discuss how the sampling model G handles the uncertainty and diversity aspects of samples. We first present an uncertainty sampling strategy by training a regressor $g^{(t)}$ in each iteration, then describe how the regularization term $\Gamma(\mathbf{v})$ is used to deal with diversity sampling.

Fig. 6.3 illustrates our sampling strategies, i.e., uncertainty sampling and diversity sampling, in comparison with random sampling. Figure 6.3.(a) describes a real data distribution with two classes (red and blue). Figure 6.3.(b) shows that random sampling can only select very few samples from the minority class (red). Figure 6.3.(c) shows using uncertainty sampling leads to samples that are similar. Figure 6.3.(d) shows that diversity sampling can evenly select samples from different groups in the sample space.

6.4.1 Uncertainty Sampling

In the LTS framework, we predict the uncertainty of samples by learning from the performance of the boosting model, i.e., the training loss. We dynamically construct a training dataset to train a regressor for predicting the uncertainty in each iteration.

Formally, a training subset $T_A^{(t)}$ for the sampling model G is constructed at the t-th iteration such that $T_A^{(t)} = \{(x_i, z_i^{(t)}) | (x_i, y_i) \in T^{(t)}, z_i^{(t)} \in [0,1]\}$, where $\mathbf{z}^{(t)} = \langle z_1^{(t)}, \ldots, z_q^{(t)} \rangle$ is generated by the softmax layer of the boosting model F and $q = |T^{(t)}|$ as shown in Eq. 6.4. Thus, each training subset $T_A^{(t)}$ contains the same set of samples as in $T^{(t)}$, but the labels of these samples in $T_A^{(t)}$ are different from the labels in $T^{(t)}$. Furthermore, each label $z_i^{(t)}$ represents the probability of being mis-classified of a sample x_i after the first t iterations. We then predict the uncertainty score $g^{(t)}(x_i)$ of an unlabeled instance $x_i \in X^{U(t)}$ in Eq. 6.5 by solving a regression problem, i.e., training $g^{(t)}$ to minimize the following objective in the t-th iteration:

$$\sum_{(x_i, z_i^{(t)}) \in T_A^{(t)}} w_i^{(t)} \ell_2(g^{(t)}(x_i), z_i^{(t)}) + \Omega_2(g^{(t)}) \tag{6.6}$$

where ℓ_2 is also a differentiable loss function, $\Omega_2(g^{(t)})$ is the penalty for the complexity of $g^{(t)}$, and $w_i^{(t)}$ is a weighted value for x_i and is dynamically adjusted during the iterations. The intuition behind $w_i^{(t)}$ is to give higher weighted values to samples that are uncertain in more iterations, rather than samples that are uncertain in fewer iterations. For example, if a sample is mis-classified by the boosting model several times, it will be assigned a higher weighted value than another sample which is mis-classified only once. We will present a method of assigning dynamic weighted values in Section 6.4.3.

6.4.2 Diversity Sampling

In the LTS framework, we deal with the diversity of samples by partitioning the sample space into a number of different groups such that samples in the same group are more similar than the samples in different groups. Then we use the regularization term $\Gamma(\mathbf{v})$ in Eq. 6.5 to regulate the sampling model, i.e., selecting samples from each group evenly.

Suppose that unlabeled samples in $X^{(t)}$ are partitioned into a set of groups $\{X_1^{(t)}, \ldots, X_b^{(t)}\}$ alike in certain features. Then we define the regularization term $\Gamma(\mathbf{v})$ over $\{X_1^{(t)}, \ldots, X_b^{(t)}\}$ using a $l_{2,1}$-norm function as:

$$\Gamma(\mathbf{v}) = ||\mathbf{v}||_{2,1} = \sum_{i=1}^{b} ||\mathbf{v}_i||_2 \tag{6.7}$$

where b is the total number of groups associated with $X^{U(t)}$, $\mathbf{v}$ is partitioned into $\{\mathbf{v}_1, \ldots, \mathbf{v}_b\}$ where $\Sigma_{i=1}^{b} |\mathbf{v}_i| = |\mathbf{v}|$, $\mathbf{v}_i \in \{0,1\}^m$, $m = |X_i^{(t)}|$ and $i \in [1, b]$. That is, $||\mathbf{v}_i||_2$ is the l_2-norm of $\mathbf{v}_i$ that is a binary vector whose elements correspond to samples in group $X_i^{(t)}$.

It is known that the $l_{2,1}$-norm favors on selecting samples with diversity [76]. When the value of the $l_{2,1}$-norm is small, non-zero entries of $\mathbf{v}$ are concentrated in a small number of groups, i.e., the distribution of samples is limited to a small number of groups and accordingly the diversity of samples is low. On the contrary, when maximizing the $l_{2,1}$-norm in Eq. 6.5, there is a counter-effect on the distribution of samples, i.e., non-zero entries of $\mathbf{v}$ are widely distributed w.r.t. as many groups as possible and thus the diversity of samples is high.

Example 6.4.1. *Consider Fig. 6.3(d) in which the sample space is partitioned into four groups*

and a number of 24 samples will be selected. If we select 6 samples from each group, $||\mathbf{v_i}||_2 = \sqrt{6}$*, we have* $\Gamma(\mathbf{v}) = \sum_{i=1}^{4} ||\mathbf{v}_i||_2 = \sqrt{6} \times 4 = 9.8$*. If we select 24 samples from only one group,* $||\mathbf{v_i}||_2 = \sqrt{24}$*, then* $\Gamma(\mathbf{v}) = \sum_{i=1}^{1} ||\mathbf{v}_j||_2 = \sqrt{24} = 4.9$*.*

6.4.3 Algorithm Description

In this section, we propose an algorithm for the LTS framework and discuss several important aspects of this algorithm which may influence the effectiveness of sampling.

Algorithm 6: Learning-To-Sample (LTS)

Input: X with k groups, i.e., $\sum_{i=1}^{k} X_i^{(0)} = X$; label budget $budget(\zeta)$; Balancing parameter α; Number of iterations n;
Output: A boosting model F

1 Initialize $T^{(0)} = \varnothing$
2 Select a set of seed samples $\Delta^{(0)}$ from k groups to maximize $\Gamma(\mathbf{v})$, where $|\Delta^{(0)}| = \frac{budget(\zeta)}{n}$
3 **for** $t = 1, \ldots, n$ **do**
4 Update $T^{(t)} = T^{(t-1)} + \Delta^{(t-1)}$
5 Train an additive function $f^{(t)}$ by minimizing the objective in Eq. 6.3 using $T^{(t)}$
6 Generate a training set $T_A^{(t)}$
7 Train a regression function $g^{(t)}$ by minimizing the objective in Eq. 6.6 using $T_A^{(t)}$
8 Update $X_i^{(t)} = \{x \in X_i^{(t-1)} | x \notin \Delta^{(t-1)}\}$, where $i = 1, \ldots, k$
9 Select a set of samples $\Delta^{(t)}$ from $\sum_{i=1}^{k} X_i^{(t)}$ by maximizing the objective in Eq. 6.5, with $|\Delta^{(t)}| = \frac{budget(\zeta)}{n}$

A high-level description of the algorithm is presented in Algorithm 6. This algorithm takes a k-grouped dataset, a label budget and the number of iterations as input. The first step is to initialize the training set T^0 and select a set of seed samples from k groups using our diversity sampling strategy (Lines 1-2). Then the algorithm iterates to train a boosting model by actively selecting samples (Lines 4-9). For each t-th iteration, we first update the training set $T^{(t)}$ by adding newly selected samples $\Delta^{(t-1)}$ into the previous training set $T^{(t-1)}$ (Line 4). Then an additive function $f^{(t)}$ is trained for the boosting model F (Line 5). After that, a new training set $T_A^{(t)}$ is generated for the sampling model G based on the output of the current F (Line 6), and a regressor is trained for uncertainty prediction (Line 7). We then update the groups $\{X_1^{(t)}, \ldots, X_k^{(t)}$ by excluding the previous selected samples in $\Delta^{(t-1)}$, and select a new set of samples $\Delta^{(t)}$ based on Eq. 6.5 Eq. 6.5 (Lines 8 - 9). The algorithm finally yields a trained boosting model as output.

In the following, we first focus on discussing three important aspects of the algorithm: (i) How to decide dynamic weighted values for samples? (ii) How to partition a sample space into different groups? (iii) How to distribute a given label budget across iterations? Then, we will discuss the softmax function used in our algorithm.

How to decide dynamic weighted values for samples?

During the training process of the boosting model, some samples in the training set may have high training losses in a number of iterations. Such samples are often informative for predicting uncertainty. Thus, a dynamic weighted value $w_i^{(t)}$ is assigned to each sample x_i to indicate its importance, as shown in Eq. 6.6. By extending the work by Freund and Schapire [53], we develop the following method of assigning dynamic weighted values in the LTS framework. In each iteration, dynamic weighted values of samples are updated in two steps:

(1) **Initialization:** For each new sample x_i at the t-th iteration, i.e., a sample in $\Delta^{(t-1)}$, we have:

$$w_i^{(t-1)} = \frac{1}{|\Delta^{(t-1)}|}. \tag{6.8}$$

(2) **Adjustment:** Then, the weighted value for each sample x_i in $T_A^{(t)}$ is re-calculated as:

$$w_i^{(t)} = w_i^{(t-1)} \times \frac{e^{-\frac{1}{2}ln(\frac{1-\epsilon^{(t-1)}}{\epsilon^{(t-1)}})g^{(t-1)}(x_i)z_i^{(t-1)}}}{Z_t}, \tag{6.9}$$

where $\epsilon^{(t-1)} = \frac{\sum_i z_i^{(t-1)}}{|T^{(t-1)}|}$ and Z_t is a normalization factor ensuring that the sum of all weighted values of samples in $T_A^{(t)}$ equals to 1.

In our algorithm, a regressor $g^{(t)}$ is iteratively trained by minimizing the objective in Eq. 6.6, in which dynamic weighted values are updated using the above method in each iteration.

How to partition a sample space into groups?

A key challenge of diversity sampling is: how to partition a sample space into groups such that samples in the same group are more similar than samples in different groups? In many real-world applications, samples that have same features are likely to be more similar than samples that have different features. Thus, we consider to partition a sample space based on available features of samples. This can also avoid common issues of sampling based on a data distribution, such as selecting too many similar samples from high density areas. In doing so, diversity sampling in our algorithm can select samples that are complementary to ones being selected by uncertainty sampling.

Formally, given a sample space with d features, a label budget $budget(\zeta)$ and a number n of iterations, we partition the sample space into k groups where $k = \lceil \sqrt[d]{\frac{budget(\zeta)}{n}} \rceil^d$ and $\lceil\ \rceil$ indicates the ceiling function. For example, if we have $budget(\zeta) = 600$, $n = 20$ and $d = 4$, then $k = \lceil \sqrt[4]{\frac{600}{20}} \rceil^4 = \lceil 2.34 \rceil^4 = 81$, i.e., 81 groups. Each of such groups corresponds to an area in the sample space and samples from the same area have some common features.

How to distribute label budget across iterations?

Under a given label budget $budget(\zeta)$, when more samples are selected at the beginning of the training process, it implies that less samples can be used in the later iterations to leverage hints from observed samples for improving performance. For example, when

$|\Delta^{(1)}| = budget(\zeta)$, i.e., all samples are used in the first iteration, the training process in the LTS framework would be the same as in the traditional training process. On the other hand, if allocating more samples to the later iterations, the boosting model F would have higher variance in the early iterations, but a better chance to "bias" samples for active learning in the later iterations.

In our algorithm, we distribute a label budget equally over all iterations, i.e., $|\Delta^{(t)}| = budget(\zeta)/n$ for any $t \in [1, n]$ (Line 2 of Algorithm 6). An alternative is to distribute samples in an exponentially decreasing manner over iterations, i.e., $|\Delta^{(t)}| = budget(\zeta)/2^t$. As will be discussed in our experiments later, the former approach outperforms the latter one in almost all cases.

Softmax function. In general, the softmax function is used to map a d-dimensional vector $\mathbf{v}$ of arbitrary real values to a d-dimensional vector $\sigma(\mathbf{v})$ of real values, where each value is in the range (0, 1), and all the values add up to 1. It can be expressed as:

$$\sigma : \mathbb{R}^d \rightarrow \{\ \sigma \in \mathbb{R}^d | 0 \leq \sigma_i \leq 1, \sum_{i=1}^{d} \sigma_i = 1\} \tag{6.10}$$

where,

$$\sigma_i = \frac{e^{v_j}}{\sum_{k=1}^{d} e^{v_k}} for\ j = 1, ..., d \tag{6.11}$$

In our approach, we only use the softmax function for a 2-dimensional vector, i.e., $\mathbf{v} = (y_i^{(t)}, \hat{y}_i^{(t)})$ for a sample x_i. Then we select the larger value of σ_i, i.e., $max\{\sigma_1, \sigma_2\}$.

6.5 Theoretical Analysis

As reported in the previous works [38; 93], the *cold start* problem often occurs in active learning because only a small amount of labeled samples is available in early iterations. Essentially, this is due to the inability of making reliable predictions by a machine learning model if training data is not sufficient. When a dataset has highly imbalanced classes (i.e., the number of samples from a majority class is much more than the number of samples from a minority class), the cold start problem can be further aggravated. Treating samples of all classes equally often leads to selecting samples that are likely to be similar or highly correlated, and thus are not representative [76; 157].

In the LTS framework, the uncertainty of samples is measured using a regressor that is dynamically trained on samples labeled with their losses from the boosting model. If we select samples by only taking the uncertainty of samples into consideration, the cold start problem would also occur in our work. Since one of the reasons underlying the cold start problem is that training data is too small to be representative, we thus partition a sample space into a number of groups based on similarity of features and introduce the regularization term $\Gamma(\mathbf{v})$ to ensure that more representative samples are selected from such a k-grouped sample space. Our experiments show that this approach works effectively for addressing the cold start problem (the experimental results will be discussed later in Section 6.6).

6.6 Experiments

We have conducted experiments to empirically verify our LTS approach, aiming to answer the following questions:

(1) Given a limited label budget, how does our LTS approach perform in comparison with other sampling methods?

(2) How effectively can our LTS approach deal with the cold start problem and the class imbalance problem?

(3) How does the balancing parameter α affect the performance of our LTS approach?

(4) How do two sampling distribution methods perform, i.e., equal distribution vs exponentially decreasing distribution?

(5) How does our LTS approach perform in reducing label budgets while still achieving the same level of quality for classification as other sampling methods?

6.6.1 Experimental Setup

We evaluate our LTS framework on widely used entity resolution datasets [135], and the results are shown in Section 6.6.2. Additionally, to test the generalization of our approach, we evaluate LTS on two additional datasets, for multi-class image classification and salary level (binary classification) prediction tasks, respectively, and the results are shown in Section 6.6.3

Datasets. Four datasets are used in our experiments: *Cora*, *DBLP-Scholar*, *DBLP-ACM* and *North Carolina Voter Registration (NCVoter)*. Details of these datasets are introduced in Section 2.2. The datasets are highly imbalanced, i.e., the number of samples from the majority class (non-match) is much more than the number of samples from the minority class (match) in these datasets.

Baselines. We use the following machine learning-based baseline methods:

- *CART* [14], short for *Classification And Regression Tree*, is a decision tree approach.
- *XG* [19], short for *eXtreme Gradient Boosting*, is a widely used and state-of-the-art boosting approach for decision trees.
- *XG+RS*, refers to applying XG on training sets built using the random sampling strategy.
- *XG+US*, refers to applying XG on training sets built only using the uncertainty sampling strategy, i.e., $\alpha = 0$ in our LTS framework.
- *XG+DS*, refers to applying XG on training sets built only using the diversity sampling strategy, i.e., $\alpha \to \infty$ in our LTS approach.

Our LTS approach is denoted as *XG+LTS*. To evaluate how the exponentially decreasing distribution of samples may affect performance, we denote a variant of *XG+LTS* as *XG+LTS(E)*

which only differs from XG+LTS in distributing samples in an exponentially decreasing manner. By default, we set $\alpha = 1$ for XG+LTS and *XG+LTS(E)*, unless otherwise stated. For XG, the maximum depth of each tree is 5, and other parameters are set as default as used in [19].

Measures. We use *precision*, *recall* and *f-measure* as measures for entity resolution instead of accuracy. Details of these measures are introduced in Section 2.2.2.

Label budgets. In our experiments, for each dataset X, we specify a label budget in terms of a certain percentage of the size of the dataset ($|X|$). For example, when using 1% as the label budget for the dataset NCVoter, i.e., 1% of $|X|$, we have 100,000 samples because NCVoter contains 10M samples in total. We also set $n = 20$ (i.e., 20 iterations), and distribute a label budget as follows:

- For the methods CART and XG, a label budget is used in the first iteration to randomly select all samples within the given label budget for training.
- For the methods XG+RS, XG+US, XG+DS and XG+LTS, a given label budget is evenly divided over 20 iterations. For example, given a label budget 1% for NCVoter, 5,000 samples are used in each iteration for 20 iterations.
- For the method XG+LTS(E), a given label budget is divided over 20 iterations in an exponentially decreasing manner.

6.6.2 Results and Discussion

We discuss our experimental results to answer the aforementioned questions at the beginning of this section.

6.6.2.1 Performance Comparison

F-measure results under different label budgets. The f-measure results under different label budgets are presented in Table 6.3. Generally, for all the datasets, all the methods converge, except CART, when the label budget is sufficient, e.g. 5% in Cora. XG+LTS outperforms all the baselines over all the datasets. The balancing parameter α for the best performance varies, depending on label budgets and datasets. For example, when the label budget is 5%, XG+LTS with $\alpha = 1$ performs best in Cora and XG+LTS with $\alpha = 0.5$ performs best in DBLP-ACM. When the label budget is relatively small, e.g. less than 1%, XG+DS achieves a better performance than XG+US in all datasets. When the label budget is larger, e.g. in the range 1% to 10%, XG+US performs better than XG+DS. In all cases, CART has the worst performance among all the methods, which is followed by XG+RS.

Additionally, the baselines CART, XG, XG+RS and XG+US have no result when the label budget is small, e.g. 0.01% in Cora and NCVoter, 0.1% in DBLP-ACM and DBLP-Scholar. However, both XG+LTS and XG+DS achieve good performance, even when the label budget is small.

Specifically, for DBLP-ACM dataset, $budget(\zeta)$ ranges from $0.5\% \times |X|$ to $10\% \times |X|$. Again, our XG+LTS algorithm outperforms all the baselines. However, different from Cora

dataset, XG + US performs the worst when $budget(\zeta)$ is small, i.e., no result generated until 2% of the dataset is used for training. For DBLP-Scholar dataset, $budget(\zeta)$ ranges from 0.01% to 10% of the entire dataset. Our XG+LTS approach has the best performance until $budget(\zeta)$ is 10% of the dataset. Similar to Cora dataset, uncertainty sampling performs well in this dataset and the best in the case of $budget(\zeta) = 10\% \times |X|$. This is because a large number of samples in this dataset have high uncertainty. For NCVoter dataset, our approach performs the best and, particularly, outperforms other baselines significantly when $budget(\zeta)$ is small, i.e., less than $1\% \times |X|$. We notice that XG+US performs worse. The reason is that samples in the majority class are similar and the uncertainty sampling fails to select samples in the minority class, i.e., having the cold start problem.

Observation 1. *From Table 6.3, we draw the following conclusions: (1) Both uncertainty sampling and diversity sampling contribute to the improvement of the performance. (2) When the label budget is limited, diversity sampling can select informative samples more effectively. However, when the label budget is sufficient, diversity samples are less informative than uncertainty samples.*

Cold start problem and class imbalance problem. Now we discuss the experimental results of our approach on dealing with the cold start problem and the class imbalance problem.

As shown in Table 6.3, when the label budget is small, i.e., 0.01% and less in Cora, 0.5% and less in NCVoter and DBLP-ACM, and 0.1% and less in DBLP-Scholar, the methods CART, XG, XG+RS and XG+US have the cold start problem (i.e, the FM values are zero). Compared with these methods, XG+LTS only has the cold start problem in the case that the label budget is 0.1% in DBLP-ACM. More interestingly, XG+DS does not have the code start problem in all settings of our experiments over all datasets. Since XG+DS is a special case of XG+LTS, this indicates that, when the label budget is small, we can handle the cold start problem by choosing a high value for the parameter α.

Since the datasets used for entity resolution are highly imbalanced. We can see from Table 6.3 that XG+DS outperforms all the other methods when the label budget is small, while all the baselines have no result. When a dataset is highly imbalanced, samples from the majority class are likely to be selected and samples from the minority class are often ignored, which aggravates the cold start problem.

Observation 2. *Our LTS framework helps to alleviate the cold start problem and the class imbalanced problem in entity resolution tasks.*

6.6.2.2 Impact of Parameters

Performance under different values of balancing parameter α. Table 6.3 shows that we have conducted experiments on different values of α (i.e., $\alpha \in \{0, 0.5, 1, 2, 5, \infty\}$) over all six datasets. When the value of α increases, the XG+LTS approach biases more on the diversity. When the label budget increases, the XG+LTS approach achieves better performance with a smaller value of α. When the budget is low, e.g. less than 0.1% in Cora dataset, a larger α has a better performance. It indicates that diversity sampling contributes more when the label budget is smaller. On the other hand, when the budget is relatively high, e.g. larger than 5% in DBLP-ACM and DBLP-Scholar, a smaller α can achieve better performance, and the f-measure results

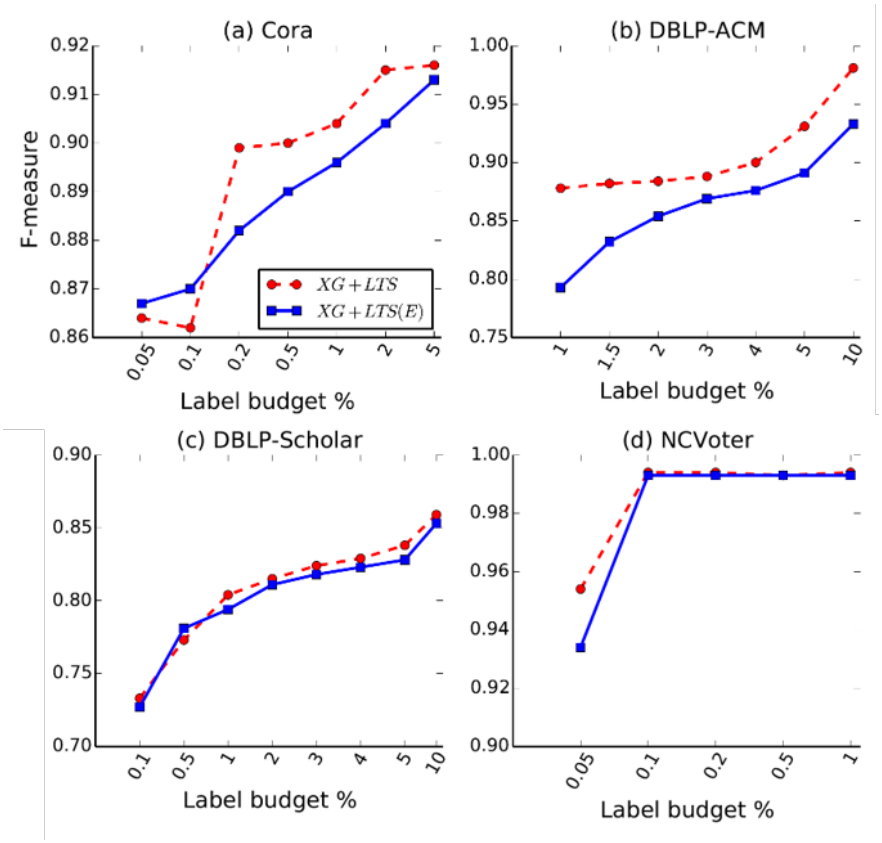

Figure 6.4: Comparison of f-measure results for the LTS approach under two different sampling distributions.

from high α is much smaller, e.g. in DBLP-ACM, the performance of $\alpha = 5$ is about 10% less than that of $\alpha = 0.5$. It indicates that uncertainty sampling contributes more when the label budget is relatively large. The f-measure results in NCVoter are not distinguishable under various values of α when the label budget is greater than 1%, since all the f-measure results are similar, i.e., larger than 0.99.

Observation 3. *The balancing parameter α affects the model performance, and it varies w.r.t. different datasets and the number of labels used.*

Performance under different sampling distribution methods. Now we discuss the experimental results of the LTS approach when using two different sampling distribution methods, i.e., XG+LTS and XG+LTS(E). The experimental results are presented in Fig. 6.4. We can see that XG+LTS obtains better f-measure results in almost all cases, except for two settings where the label budgets are very small: 0.01% in Cora and 0.1% in DBLP-Scholar. This is due to that diversity sampling contributes more in these cases. Therefore, in our LTS approach, we choose equal sampling distribution rather than exponentially decreasing sampling distribution.

Furthermore, in Fig. 6.4, comparing two sampling distribution strategies, i.e., equally distributed (denoted as XG+LTS) and exponentially decreasing distributed (denoted as XG+LTS(E)),

we find that XG+LTS gains more stable performance than XG+LTS(E) when the label budget is small. Especially in the NCVoter dataset, a limited number of samples may lead to no result as output, e.g. $budget(\zeta) < 0.05\%$. However, when the label budget is large, e.g. 2% of Cora dataset, XG+LTS(E) obtains more stable performance, which is 0.045 vs 0.018, but the difference is not relatively significant. The SD values converge to zero when the label budget is sufficiently large. The reason is that in the early stage, the number of samples is smaller than the number of groups. Although the samples are selected from each group, it is still likely to select samples with less informative. Since seed sample plays a very important role in active learning, generally speaking, an equally distributed number of samples for each iteration is enough in our framework when the sample size is limited.

Observation 4. *The equal distribution strategy works better in most of the datasets under different label budget. It is the reason why we choose this strategy as the default strategy in our experiments.*

Table 6.1: Comparison of label budgets w.r.t. classification results with desired FM values, where XG+LTS has $\alpha = 1$.

Dataset	Cora	DBLP-ACM	DBLP-Scholar	NCVoter
CART	5%	10%	10%	3%
XG	4%	8%	2%	2%
XG + RS	5%	12%	5%	2%
XG + US	2%	7%	2%	7%
XG + DS	3%	10%	2%	**0.03%**
XG + LTS	**0.5%**	**4%**	**0.9%**	**0.03%**
FM values	0.9	0.9	0.8	0.9

6.6.2.3 Label Efficiency

Table 6.1 presents our experimental results on the label cost under the same performance. We set the desired FM value as 0.9 for each dataset, except for the dataset DBLP-Scholar. This is because the dataset DBLP-Scholar is noisy and a classification result with the FM value 0.9 can hardly be achieved. Therefore, we set the desired FM value 0.8 for this dataset. Then we record the amount of label budgets required by each method in order to achieve the desired F-measure values. From Table 6.1, we can see that, our XG+LTS method ($\alpha = 1$) requires the smallest number of samples for each of these datasets, in comparison with the other baseline methods. Especially, for the dataset NCVoter, our XG+LTS approach requires a significantly smaller number of samples for achieving the same performance, in comparison with the baseline methods CART, XG, XG+RS and XG+US. Although XG+DS requires a comparable label budget as our XG+LTS method for the dataset NCVoter, it requires at least a double amount of label budgets for the other three datasets.

Observation 5. *Our LTS framework can reduce the label cost significantly for all datasets while still remaining a comparable performance w.r.t. all the baselines.*

6.6.3 Supplementary Experiments on Classification Tasks

Table 6.2: Datasets for Classification Tasks.

Classification Tasks	Datasets	# Attributes	# Samples ($\|X\|$)	Types of Labels
Image classification	Mnist	28×28	60,000	10 digits (i.e., 0-9)
Salary level prediction	Adult	14	48,842	$\{> 50k, \leq 50k\}$

In this section, we conduct more experiments on traditional classification tasks, i.e., image classification and salary level prediction to show that our model is not limited for ER tasks.

Two datasets are used in the experiments, details are shown in Table 6.2: (1) *Mnist*[1] dataset contains 28×28 images, and each image corresponds to a handwritten digit. The task is to classify the images into ten categories, i.e., from 0 to 9. (2) *Adult*[2] dataset contains adults' personal information. The task is to predict if a person's salary income is more than 50k.

Fig. 6.5 presents the performance (accuracy) of our approach and the baseline methods. For the dataset Mnist, both XG+US and XG+LTS obtain better results than the others. The reason why XG+DS does not perform well is due to the large feature space of Mnist. There are in total 784 features in this dataset. Thus, the number of groups is much larger than the number of samples being selected in each iteration, which leads to suboptimal performance. For the dataset Adult, XG+DS performs better than XG+US when the label budget is limited, e.g. less than 0.2%. However, XG+US achieves better performance when the label budget increases, e.g. more than 1%.

6.7 Summary

In this chapter, we have proposed a novel learning-based active learning framework called Learning-To-Sample. This framework is composed of a sampling model G and a boosting model F. The boosting model contains a dynamic training set. A subset of samples are added into this training set in each iteration. These additional samples are selected iteratively by the sampling model which can learn from the performance of the boosting model through a unified process for two sampling strategies: uncertainty sampling (US) and diversity sampling (DS). The experimental results show that our approach outperforms all the baselines, particularly when the number of samples is limited. In addition to this, our framework can handle the cold start problem and the class imbalance problem.

[1] Available from: *http://yann.lecun.com/exdb/mnist/*
[2] Available from: *https://archive.ics.uci.edu/ml/datasets/adult*

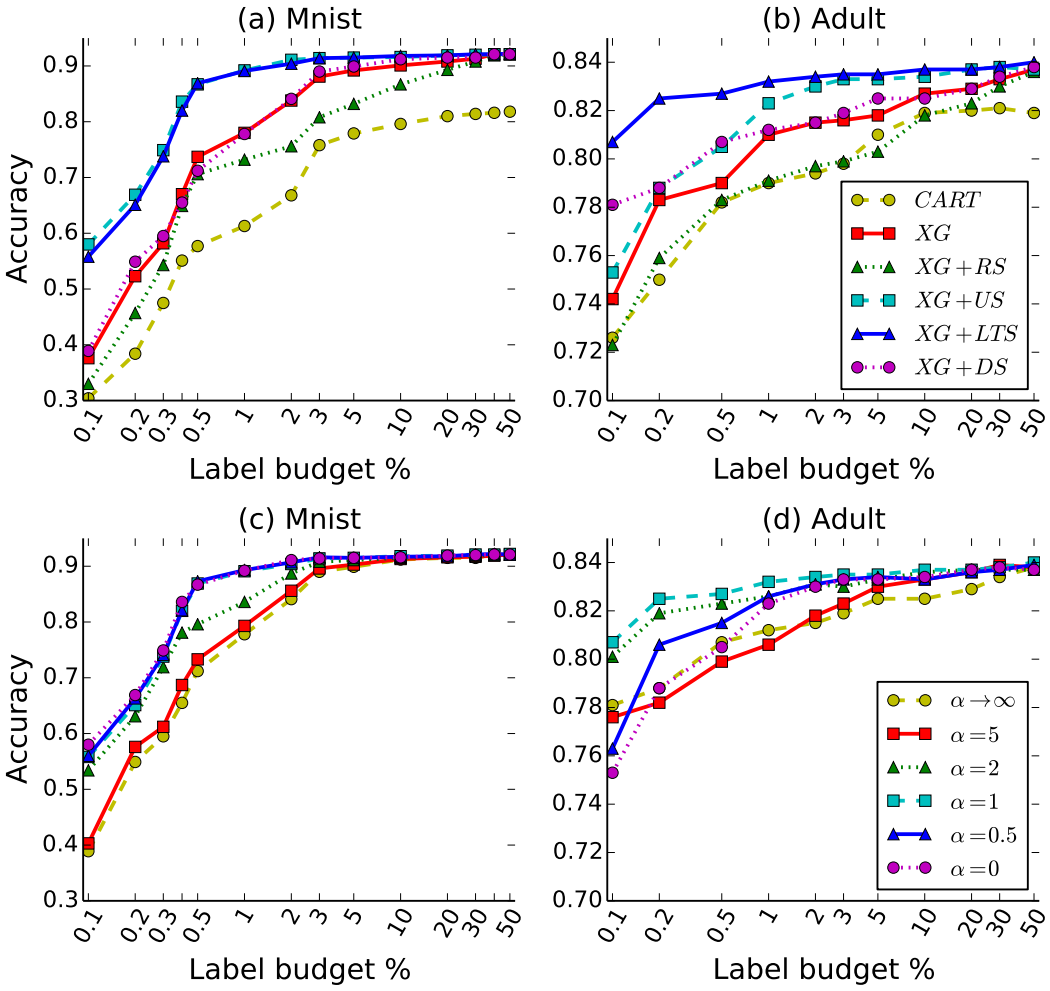

Figure 6.5: Comparison of accuracy results for image classification and salary level prediction tasks under different label budgets.

Table 6.3: Comparison of f-measure results for entity resolution tasks under different label budgets.

Dataset	Label Budget $budget(\zeta)$ (% of $\|X\|$)	CART	XG	XG+RS	XG + US $\alpha = 0$	XG+LTS $\alpha = 0.5$	XG+LTS $\alpha = 1$	XG+LTS $\alpha = 2$	XG+LTS $\alpha = 5$	XG + DS $\alpha \to \infty$	XG + LTS(E) $\alpha = 1$
Cora	0.01	0	0	0	0	0.637	0.857	0.861	0.867	**0.878**	0.862
	0.05	0.741	0.763	0.750	0.827	0.851	0.864	0.870	0.883	**0.885**	0.867
	0.1	0.788	0.796	0.787	0.823	0.863	0.862	0.873	**0.887**	0.886	0.870
	0.5	0.848	0.835	0.835	0.873	0.893	**0.900**	0.895	0.895	0.893	0.890
	1	0.868	0.878	0.880	0.870	0.896	0.902	**0.904**	0.898	0.894	0.896
	5	0.878	0.897	0.892	0.907	0.912	**0.915**	0.913	0.902	0.898	0.904
NCVoter	0.01	0	0	0	0	0.403	0.324	0.403	0.752	**0.875**	0.571
	0.05	0	0	0	0	0.903	0.954	0.989	**0.993**	0.991	0.934
	0.1	0	0	0	0	0.989	**0.994**	0.993	0.993	0.993	0.993
	0.5	0	0	0	0	0.993	**0.994**	0.993	0.993	0.991	**0.994**
	1	0.334	0.379	0.398	0	0.993	0.993	0.993	0.992	**0.994**	0.993
	5	0.993	0.993	0.994	0.993	0.993	**0.997**	0.993	0.994	0.993	0.994
DBLP-ACM	0.1	0	0	0	0	0	0	0	0	**0.397**	0
	0.5	0	0	0	0	0.382	0.702	**0.720**	0.651	0.632	0.679
	1	0.348	0.347	0.279	0	0.813	**0.878**	0.778	0.730	0.721	0.793
	2	0.599	0.767	0.680	0.403	0.851	**0.884**	0.867	0.789	0.783	0.854
	5	0.870	0.850	0.803	0.874	**0.935**	0.931	0.889	0.837	0.833	0.891
	10	0.903	0.911	0.890	0.926	**0.983**	0.981	0.937	0.893	0.899	0.933
DBLP-Scholar	0.1	0	0	0	0	0.586	0.723	0.733	**0.741**	0.731	0.727
	0.5	0.378	0.54	0.498	0.555	0.764	0.773	**0.794**	0.790	0.780	0.781
	1	0.562	0.669	0.659	0.738	0.793	0.804	**0.808**	0.793	0.792	0.794
	2	0.772	0.806	0.771	0.807	0.810	**0.815**	0.813	0.799	0.801	0.811
	5	0.773	0.822	0.803	0.836	**0.838**	0.836	0.831	0.821	0.818	0.828
	10	0.808	0.835	0.830	**0.865**	0.859	0.851	0.844	0.837	0.829	0.853

Chapter 7

Generative Adversarial Networks for Entity Resolution

7.1 Introduction

Generative adversarial network (GAN) and its variants have recently emerged as a powerful deep learning technique for real-world applications across various domains such as image generation and natural language processing [60; 59]. Inspired by these advances, in this chapter, we develop a novel semi-supervised generative adversarial network, called ERGAN, to solve the aforementioned challenges faced by entity resolution applications. In ERGAN, there are two key components, which are optimized in an adversarial learning manner: (1) a *label generator* G that aims to generates pseudo labels for unlabeled samples, and (2) a *discriminator* D that aims to distinguish samples with pseudo labels from samples with real labels. The discriminator D is trained using not only a small number of samples with real labels but also a large number of samples with high-quality pseudo labels.

However, the question arises: how to ensure high-quality pseudo labels generated for unlabeled samples? Unfortunately, the existing GAN and its variants cannot guarantee this when the number of samples with real labels is limited. To address this question, our model ERGAN is designed to incorporate two modules: *diversity module* and *propagation module* into the label generator G and the discriminator D, respectively. The diversity module enriches the diversity of unlabeled samples during the sampling process, while the propagation module guarantees that only unlabeled samples with high-quality pseudo labels can be propagated into the training of the discriminator D. Consequently, even when only a very limited number of labeled samples are available, ERGAN can still effectively infer the true distribution of all labels in a semi-supervised manner. We theoretically prove that ERGAN overcomes the model collapse and convergence problems in the original GAN. Specifically, the following properties of ERGAN are proven: (1) The global optimality can be guaranteed; (2) The label generator G and the discriminator D converge to the equilibrium point; (3) The diversity module helps improve the generalization without compromising the global optimality and equilibrium; and (4) The mode collapse issue is alleviated in ERGAN. The experimental results show that ERGAN outperforms all state-of-the-art baselines, including unsupervised, semi-supervised and full-supervised learning methods. Even when the label cost is very limited and the state-of-the-art baselines fail to predicate labels, ERGAN can still achieve reasonably good performance.

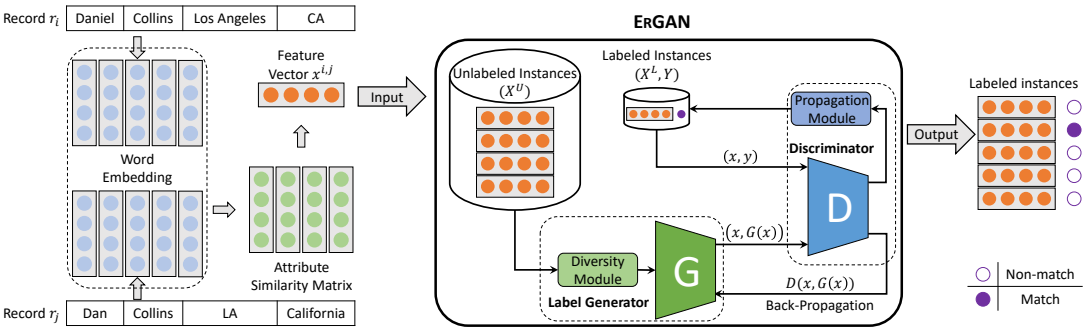

Figure 7.1: Overview of the ERGAN framework. Using only a limited number of labeled samples for training, ERGAN takes unlabeled samples as input and classifies them as being matches or non-matches (i.e., predicting their labels).

Figure 7.1 shows an overview of our framework ERGAN for entity resolution tasks. Given a record pair r_i and r_j, each record is first represented as a matrix of vector embeddings using word embedding. Based on this, the record pair is transformed into an attribute similarity matrix, which leads to a feature vector (i.e., an unlabeled sample). Takes these feature vectors (as unlabeled samples), together with a limited number of labeled samples, as input, the ERGAN will finally generate all samples with labels.

It is worthy to note that, although we only consider the use of ERGAN for entity resolution in this chapter, the techniques of ERGAN for handing overfitting and imbalanced data can be much more widely applicable. Additionally, our ERGAN framework can be used jointly with any word embedding or string matching techniques for entity resolution tasks.

The rest of this chapter is structured as follows. Section 7.2 introduces the notations used in the chapter and the formulation of the entity resolution classification problem. Section 7.3 presents the ERGAN framework which contains two components: a label generator and a discriminator, as well as the diversity module and the propagation module for each component, respectively. Section 7.4 analyzes some properties of ERGAN including the diversity of samples, the global optimality, the equilibrium and the mode collapse. Section 7.5 discusses our experimental results. We conclude the chapter in Section 7.6.

7.2 Problem Formulation

Let R be a dataset consisting of a set of records where each $r \in R$ is associated with a number of attributes A. We use $r.a$ to refer to the value of an attribute $a \in A$ in a record r. Each record pair (r_i, r_j) in R corresponds to a feature vector $x^{(ij)} \in \mathbb{R}^m$ where each dimension $x_k^{(ij)}$ indicates a feature value, e.g., the textual similarity of values in an attribute $a_k \in A$, i.e., $x_k^{(ij)} = f(r_i.a_k, r_j.a_k)$, calculated by a function f.

Let $X = \{x^{(ij)} | (r_i, r_j) \subseteq R \times R\}$ be the set of all samples corresponding to record pairs in

R and $Y = \{M, N\}$ be a label space where M and N refer to two labels *match* and *non-match*, respectively. There is a small subset $X^L \subseteq X$ of samples that are labeled, while the other samples in X are unlabeled, i.e., $X^U = X - X^L$. We assume $|X^L| \ll |X^U|$, i.e., X has a very limited number of labeled samples in X^L but a large number of unlabeled samples in X^U. We denote (X^L, Y) as a set of samples in X^L and their labels in Y, and $(x^L, y) \sim (X^L, Y)$ as a pair of sample $x^L \in X^L$ and its label $y \in Y$. Our task is to tackle the entity resolution classification problem as formulated below.

Definition 9. *Given a set X of samples with $X = X^L \cup X^U$ and $|X^L| \ll |X^U|$, and a label space $Y = \{M, N\}$, the **entity resolution classification problem** is to learn a model Λ that can predict a label $\hat{y} \in Y$ for each unlabeled sample $x \in X^U$ w.r.t.*

$$max\ E(\Lambda)/|X^U| \tag{7.1}$$

where $E(\Lambda) = \sum_{x \in X^U \wedge \hat{y} = y} 1.$

Intuitively, $E(\Lambda)$ refers to the total number of unlabeled samples in X^U whose labels are correctly classified by Λ.

7.3 Proposed Method: ErGAN

Our proposed method ERGAN consists of two components: (1) a *label generator* G; and (2) a *discriminator* D. Both G and D are differentiable functions.

7.3.1 Label Generator

In ERGAN, a label generator G can obtain samples from $p(X^U)$, but does not know about $p(Y)$ nor $p(X, Y)$. Nevertheless, we know that $p(X^U) \approx p(X)$ because $X^U \subseteq X$ and $|X^U|/|X|$ is close to 1. The goal of G is to learn a conditional distribution $p_g(Y|X^U) \approx p(Y|X^U)$, i.e., given an sample $x \sim p(X^U)$ as input, G generates a pseudo label $\hat{y}$ for x. Ideally, the pseudo label $\hat{y}$ generated for an sample x by G should be the same as the real label of x. To simulate the conditional distribution $p(Y|X^U)$, the label generator G receives feedback (i.e., gradients) from the discriminator D and is trained iteratively through backpropagation.

Diversity module. One major difference of our ERGAN from the original GAN and its variants such as CatGAN [138] is that we consider the diversity of samples in the minibatch sampling process. More specifically, for all samples in X, we partition them into a number of non-overlapping subspaces alike in certain features $\{X_1, \ldots, X_b\}$ such that samples in the same subspace are more similar than those in different subspaces. Accordingly, labeled samples in X^L and unlabeled samples in X^U are partitioned into these b subspaces, i.e., $X_i^L = X_i \cap X^L$ and $X_i^U = X_i \cap X^U$.

Let $\mathbf{v} = (\mathbf{v}_1, ..., \mathbf{v}_b)$ be a binary vector corresponding to b subspaces, where each $\mathbf{v}_i = (v_i^1, \ldots, v_i^{n_i})^T \in [0, 1]^{n_i}$ and $n_i = |X_i^U|$. That is, each v_i^j $(1 \leq j \leq n_i)$ is associated with an sample in X_i^U. Then, a minibatch of m samples is selected from X^U according to the following

objective function:

$$\text{maximize} \qquad ||\mathbf{v}||_{2,1} \quad \text{s.t.} \sum_{i,j} v_i^j = m \tag{7.2}$$

where $||\mathbf{v}||_{2,1}$ is a $l_{2,1}$-norm function defined as:

$$||\mathbf{v}||_{2,1} = \sum_{i=1}^{b} ||\mathbf{v}_i||_2 = \sum_{i=1}^{b} \sqrt{\sum_{j=1}^{n_i} v_i^{j2}} \tag{7.3}$$

Here, $||\mathbf{v}_i||_2$ is the l_2-norm of $\mathbf{v}_i$. When $v_i^j = 1$, the sample in X_i^U corresponding to v_i^j is selected into the minibatch; otherwise, that sample is not selected. When the value of the $l_{2,1}$-norm is small, samples are selected from a small number of subspaces in X^U and the diversity of samples is low. Conversely, when maximizing the $l_{2,1}$-norm in Eq. 7.2, samples are selected from as many subspaces in X^U as possible and the diversity of samples is high.

Objective function of G. After a minibatch of unlabeled samples is selected from X^U according to Eq. 7.2, the label generator G generates a pseudo label $G(x_i)$ for each unlabeled sample x_i in the minibatch. Then, $(x_i, G(x_i))$ is sent to the discriminator D. After receiving the gradient from D, G updates its parameters according to the following objective:

$$\mathcal{L}_G = \min_G \quad \mathbb{E}_{x \sim p(X_i^U)}[\log(1 - D(x, G(x)))] \tag{7.4}$$

7.3.2 Discriminator

Unlike GAN, a discriminator D in our ERGAN does not know about the real distribution $p(X, Y)$. Instead, D has access only to a limited number of samples with real labels, i.e., (X^L, Y). The goal of D is to distinguish whether a labeled sample $(x, G(x))$ is from the real distribution $p(X, Y)$, i.e., given a pair $(x, G(X))$ as input, D generates a scalar value in $[0, 1]$ to indicate the probability that $G(x)$ is the same as the real label y of x.

Propagation module. To achieve the above goal, as opposite to GAN and its variants in which the discriminator has the true distribution $p(X, Y)$, D in ERGAN is designed to approximate the true joint distribution $p(X, Y)$ progressively through a propagation module. The general principle of propagation is that, the more confident the pseudo label $G(x)$ of an sample x is the same as its real label y, the more likely such an sample is selected. Specifically, let $(X^t, G(X^t))$ denote all unlabeled samples with their pseudo labels at the t-th iteration of propagation. These samples are fed to D to obtain their scores $D(X^t, G(X^t))$ that indicates the probabilities of their pseudo labels being the same as their real labels. Based on the scores, a subset $\Delta X^t \subseteq X^t$ of samples is selected according to the following objective function:

$$\begin{aligned} &\underset{\Delta X^t \subseteq X^t}{\text{argmax}} \sum_{x \in \Delta X^t} D(x, G(x)) \\ &\qquad \text{subject to } |\Delta X^t| = \gamma \end{aligned} \tag{7.5}$$

where γ is a hyper-parameter for the number of unlabeled samples being selected in the t-th iteration of propagation.

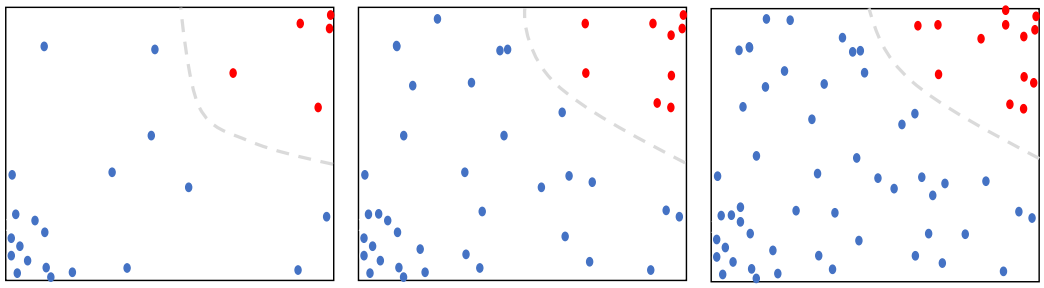

Figure 7.2: An illustration for propagation of ERGAN. A boundary between two classes (red and blue) is learned through propagation.

Then, this subset of samples with their high-quality pseudo labels $(\Delta X^t, \hat{Y})$ is propagated into the set of labeled samples $(X^*, Y)^t$ to train D, i.e.,

- $(X^*, Y)^0 = (X^L, Y)$
- $(X^*, Y)^t = (X^*, Y)^{t-1} \cup (\Delta X^t, \hat{Y})$

Hence, at the t-th iteration of propagation, D has access to $(X^*, Y)^t$, which is a mixed set of labeled samples from X^L (with real labels) and unlabeled samples from X^U (with pseudo labels generated by G). The following holds:

$$(X^*, Y)^0 \subseteq (X^*, Y)^1 \subseteq \cdots \subseteq (X^*, Y)^t \tag{7.6}$$

Fig. 7.2 shows an example of the propagation in two iterations, where the grey dash line indicates a boundary between two classes (red and blue) and is learned through propagation.

Objective function of D. The objective function of D at the t-th iteration of propagation is defined as:

$$\begin{aligned} \mathcal{L}_D = \max_D \quad & \mathbb{E}_{x \sim p(X_i^U)} \log[(1 - D(x, G(x)))] \\ & + \lambda \mathbb{E}_{(x,y) \sim (X^*, Y)^t} \log[D(x, y)] \end{aligned} \tag{7.7}$$

where λ refers to a weighted term. In the following, we will explain how unlabeled samples with their pseudo labels, i.e., $(X^t, \hat{Y})$, is selected at the t-th iteration of propagation.

7.3.3 Algorithm Description

Our algorithm is described in Algorithm 7. It involves two key processes: batch training (Lines 3-8) and label propagation (Lines 9-12). In the batch training process, a minibatch is randomly selected from unlabeled samples in X^U and G generates pseudo labels for samples in this minibatch (Lines 4-5). Then another minibatch is randomly selected from samples in $(X^*, Y)^t$ (Line 6). After that, the discriminator D is trained using these two minibatches w.r.t. Eq.

7.7, while the label generator G is trained w.r.t. Eq. 7.4. The label propagation process only occurs after each batch training is finished. At this time, G and D reach (or closely approach) the equilibrium point and their parameters are fixed. In the label propagation process, G first generates pseudo labels for all unlabeled samples in X^t, and then D predicts scores for these samples. Based on these scores, a subset of samples $\Delta X^t \subseteq X^t$ with high-quality pseudo labels is propagated into the set of labeled samples $(X^*, Y)^t$ for D in the t-th iteration.

How to choose b, t and n? In our algorithm, the number of subspaces b is decided based on the attributes in each dataset. Suppose that a dataset has four attributes, we first obtain the median value for each attribute, and then partition samples into $4^2 = 16$ subspaces according to whether attribute values of each sample are above or below the median values of these four attributes [136]. Furthermore, n is a hyper-parameter referring to the number of iterations for converging G and D, and t is decided by the total number X^U of unlabeled samples and the number γ of samples being propagated in each iteration, i.e., $t = \lceil \frac{|X^U|}{\gamma} \rceil$.

Algorithm 7: Minibatch stochastic gradient descent and label propagation of ERGAN

Input: b subspaces in X; X^U; (X^L, Y);
Output: $(X^*, Y)^t$ where $X^* = X$

Initialize $t = 0$; $(X^*, Y)^0 = (X^L, Y)$; $X^0 = X^1 = X^U$
while $X^t \neq \varnothing$ **do**
  for n iterations **do** `// Batch training`
    Sample a minibatch $\{x_1, ..., x_m\}$ from X^U w.r.t. Eq. 7.2
    Generate pseudo labels $\{(x_1, G(x_1)), ..., (x_m, G(x_m))\}$
    Sample a minibatch $\{(x_1^L, y_1), ..., (x_m^L, y_m)\}$ from $(X^*, Y)^t$
    Update the parameters of D w.r.t. Eq. 7.7
    Update the parameters of G w.r.t. Eq. 7.4
  t=t+1 `// Label propagation`
  Generate pseudo labels for X^t
  Select $\Delta X^t \subseteq X^t$ for propagation w.r.t. Eq. 7.5
  $(X^*, Y)^t = (X^*, Y)^{t-1} \cup (\Delta X^t, \hat{Y})$; $X^{t+1} = X^t - \Delta X^t$

7.4 Theoretical Analysis

We prove several nice theoretical properties of ERGAN.

Diversity. We first show that partitioning X into subspaces for diversity does not affect the learning goals of G and D.

Lemma 7.4.1. *If $p_g(Y|X_i^U)$ by G approximates $p(Y|X_i^U)$ for each feature subspace X_i^U, then $p_g(Y|X^U) \approx p(Y|X^U)$.*

Proof. X^U is partitioned into non-overlapping subspaces $\{X_i^U\}_{i=1}^b$. Thus, $p(X^U)$ is composed of b distributions $p(X_i^U)(1 \leq i \leq b)$. Further, Eq. 7.2 prefers samples from different subspaces, but within each subspace, samples are selected randomly. □

Since unlabeled and labeled samples are selected from the corresponding subspaces, we have the following lemma for D.

Lemma 7.4.2. *If $p_d(X_i^U, Y)$ by D approximates $p(X_i^U, Y)$ for each feature subspace X_i^U, then $p_d(X^U, Y) \approx p(X^U, Y)$.*

By $|X^L| << |X^U|$, we have $p(X^U) \approx p(X)$. Accordingly, $p_g(Y|X^U) \approx p_g(Y|X)$, $p_d(X^U, Y) \approx p_d(X, Y)$, and $p(X^U, Y) \approx p(X, Y)$.

Global optimality. Here we use $p_d(X^*, Y)^t$ to refer to the distribution learned by D at the t-th iteration of propagation. We have the following lemma.

Lemma 7.4.3. *For the fixed label generator G, the optima D in the t-th iteration of propagation is:*

$$D_G^*(x, y) = \frac{p_d(X^*, Y)^t}{p_d(X^*, Y)^t + p_g(X, Y)^t} \tag{7.8}$$

Proof. At the t-th iteration of propagation, by Eq. 7.7, the training objective of D is to maximize:

$$\mathbb{E}_{x \sim p(X_i^U)} \log[(1 - D(x, G(x)))] + \lambda \mathbb{E}_{(x,y) \sim p_d(X^*, Y)^t} \log[D(x, y)]$$

Based on the "change of variable" technique [57], we have $p_g(s) = p_{X_i^U}(G^{-1}(y)) \frac{dG^{-1}(y)}{d(s)}$, where $s = (x, y)$ denotes the vector concatenated by x and y at the t-th iteration of propagation. The first part of the above formula equals to:

$$\begin{aligned} &\int_s p_{X_i^U}(G^{-1}(y)) \log(1 - D(s)) dG^{-1}(y) \\ =&\int_s p_g(s) \log[(1 - D(s))] d(s) \end{aligned}$$

Hence, we have an objective function that is the same as GAN [60]. This lemma is proven. □

As a result, we have the following lemma.

Lemma 7.4.4. *The global minimum of the training criterion of G under the optimal D at the t-th iteration of propagation is achieved when $p_d(X^*, Y)^t = p_g(X, Y)^t$.*

Equilibrium. By Lemma 7.4.4, we have $p_g(X, Y)^t = p_d(X^*, Y)^t$. Thus, the equilibrium of the minimax game in ERGAN is achieved at each iteration of propagation. When the labels are propagated sufficiently, the real data distribution is well simulated. The following lemma states that D can approximate the real distribution $p(X, Y)$ when the number of propagation iterations is large.

Lemma 7.4.5. *$p_d(X^*, Y)^t$ approaches $p(X, Y)$ when t increases.*

Statistically, according to the *Central Limit Theorem*, the larger sample size the more likely an estimated distribution is close to the real distribution. This lemma corresponds to the central property of self-training based semi-supervised learning approaches [61; 143].

Mode collapse. The mode collapse problem occurs in GAN where the generator collapses to generate limited and nonsensical images. However, our approach ERGAN does not suffer from this problem for two reasons: (1) ERGAN takes unlabeled samples as input for both G and D. This allows D to distinguish $p_d(X^*, Y)^t$ from $p_g(X, Y)^t$, and D can prevent G to generate the same kind of pseudo labels for unlabeled samples, i.e., mode collapse. (2) In GAN, mode collapse only occurs when the discriminator is well trained but the generator is not optimal. In ERGAN, $(X^*, Y)^t$ only has limited samples with labels when t is small. Thus, D can hardly be optimal in early iterations of propagation. For later iterations, since G is trained iteratively based on its previous parameters, the mode collapse issue can also be avoided.

Overfitting problem and imbalanced class problem. ERGAN can alleviate the overfitting problem. This is because, instead of training a single classifier, ERGAN trains the label generator G and the discriminator D adversarially such that D improves G to generate pseudo labels for unlabeled samples and G regularizes D from overfitting through propagating unlabeled samples with high-quality pseudo labels. In essence, G serves as an adaptive regularization term acting on D during the minibatch training process. ERGAN can also alleviate the imbalanced class problem. We observe that, the key to solving the imbalanced class problem lies in how to effectively approximate the true distribution using limited samples with real labels. Driven by this observation, ERGAN employs the diversity module to select samples diversely from different feature subspaces. This leads to a dual boosting: both improving label efficiency of selecting samples from the minority class, and improving learning efficiency of learning a fast approximation on the underlying data distribution.

7.5 Experiments

We evaluate ERGAN to answer the following questions:

Q1. How does ERGAN perform in comparison with the state-of-the-art unsupervised, semi-supervised and fully supervised methods?

Q2. How do the design choices such as the diversity module, the propagation module, and GAN architecture affect performance of ERGAN?

Q3. To what degree ERGAN can work for classifying samples when the label cost is extremely limited (i.e., only a small number of samples with real labels)?

7.5.1 Experimental Setup

Datasets. We use four widely used benchmark datasets of entity resolution tasks: *Cora*, *DBLP-Scholar*, *DBLP-ACM* and *North Carolina Voter Registration (NCVoter)*. Table 2.2 summarizes the characteristics of these four datasets which are highly imbalanced.

Baselines. We compare ERGAN with the following baselines:

- *Unsupervised methods: Two-Steps* (*2S*) is a widely-used unsupervised learning method proposed by Christen [24]. *Iterative Term-Entity Ranking and CliqueRank* (*ITER-CR*) is the state-of-the-art graph based unsupervised method for entity resolution [158].
- *Semi-supervised methods: Semi-supervised Boosted Classifier* (*SBC*) is the state-of-the-art semi-supervised learning method with label propagation based on Adaboost classifier [129] proposed by Kejriwal and Miranker [87].
- *Fully supervised methods: Magellan* is a state-of-the-art open-source fully supervised learning-based entity resolution solution designed by Konda et al. [92]. We consider two supervised classifiers provided in Magellan: *Logistic Regression (LR)* and *Support Vector Machine (SVM)*. *eXtreme Gradient boosting* (*XGboost*) is a state-of-the-art fully supervised ensemble learning-based method proposed by Chen and Guestrin [19].
- *Deep learning-based methods: DeepMatcher* (*DM*) is a state-of-the-art deep learning-based entity matching method for entity resolution [113]. *Deep Transfer active learning* (*DTAL*) is the state-of-the-art active learning method which combines both transfer learning and active learning for handling entity resolution tasks [81].

To make a fair comparison, we follow the default parameters suggested in the original papers of the baselines. Note that DM uses the imbalance rate as a hyper-parameter which is normally unknown in real-life applications. Both DTAL and DM are deep learning-based methods in which FastText [12] is used for learning word embeddings. For other baselines, we use 2-gram Jaccard similarity for textual comparison.

To compare with the baselines that use word embeddings, we use **ERGAN+WE** to refer to the model of ERGAN augmented with word embeddings for attribute values. In our ablation study, we use **ERGAN-D** and **ERGAN-P** to refer to a model being obtained by removing the diversity and propagation modules from ERGAN, respectively, and **ERNN** a model in which the GAN architecture (i.e., G and D are trained alternatively) is replaced by a single multi-layer perceptron for semi-supervised learning with the diversity module. We set $\lambda = 1$, $m \leq 100$, and $\gamma = |X^*|$ at each iteration of propagation. Our models use the same word embedding and similarity comparison techniques as the baselines.

Measures. We use the following widely used measures in entity resolution tasks for performance evaluation: *Recall*, *Precision* and *F-measure(FM)*. Details of these measures are introduced in Section 2.2.2. Due to the existence of highly imbalanced classes in entity resolution datasets, F-measure is preferred rather than accuracy in our experiments.

7.5.2 Results and Discussion

In this section, we discuss the results of our experiments to answer the aforementioned questions.

7.5.2.1 Performance Comparison

Performance comparison under 60% training set. We conduct an experiment to evaluate how our methods perform against the baselines. Following the previous work for the super-

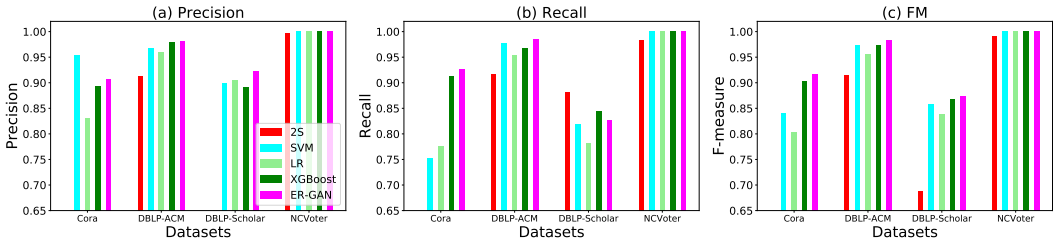

Figure 7.3: Comparison of precision, recall and f-measure results with 20% training over four datasets.

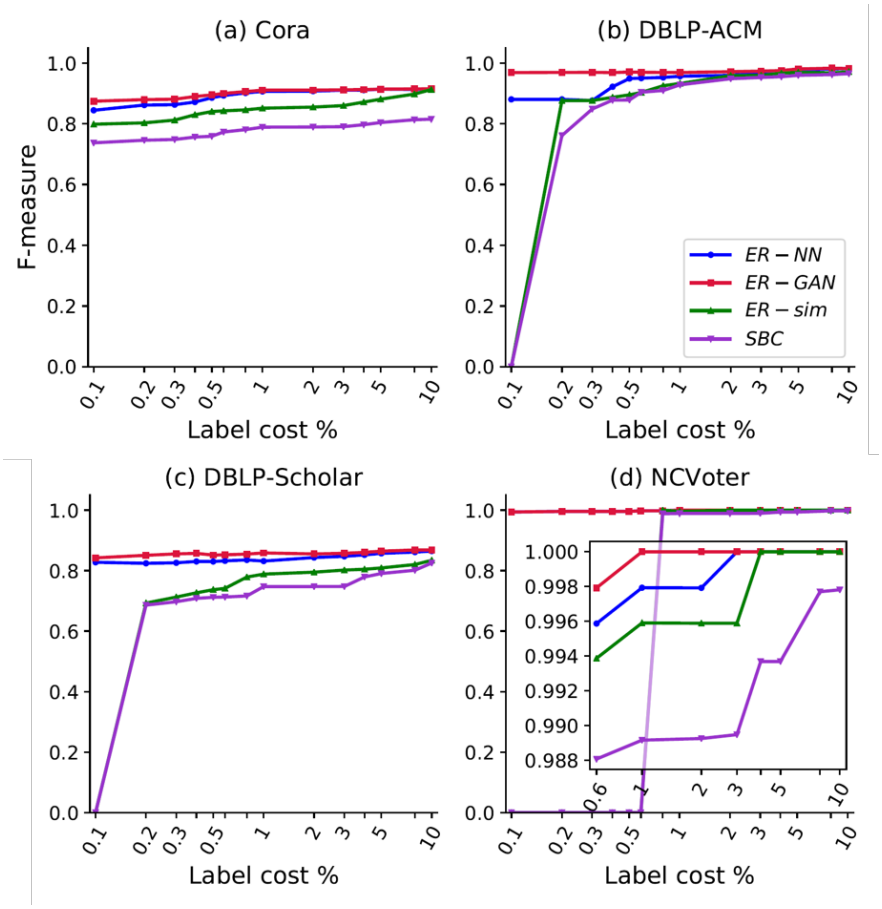

Figure 7.4: Comparison of f-measure results with 0.1% – 10% training for ablation study under four datasets.

Table 7.1: Comparison of f-measure results with 60% training. The results marked by $*$ are taken from the original chapters and the others are obtained by running the code provided by the authors.

Method	Datasets			
	Cora	DBLP-ACM	DBLP-Scholar	NCVoter
2S [24]	62.69	91.43	68.78	98.96
ITER-CR* [158]	89.00	–	–	–
SBC [87]	85.71	97.09	85.47	99.78
SVM [92]	88.95	97.19	85.71	98.48
LR [92]	80.25	95.56	83.84	99.37
XGBoost [19]	91.34	97.20	86.63	**100**
ERGAN	**93.03**	**98.23**	**88.32**	**100**
DM [113]	98.58	98.29	94.68	**100**
DTAL* [81]	$98.68_{\pm 0.26}$	$98.45_{\pm 0.22}$	$92.94_{\pm 0.47}$	–
ERGAN+WE	$\mathbf{98.72}_{\pm 0.15}$	$\mathbf{98.51}_{\pm 0.23}$	$\mathbf{94.73}_{\pm 0.35}$	**100**

vised methods DM [113] and DTAL [81], we split the datasets with 60% for training and the rest for testing.

Table 7.1 shows the results of the experiment, where the last three methods DM, DTAL and ERGAN+WE are deep-learning methods which use word embeddings for attribute values and the other methods use Jaccard similarity for comparing attribute values (without using word embeddings). We can see that, the unsupervised method 2S performs the worst among all the methods. However, the other unsupervised method ITER-CR performs better than SBC, SVM and LR due to its ability to leverage graph based structure. Compared with the fully supervised methods, the semi-supervised method SBC performs better than LR, comparably with SVM, but worse than XGBoost. Our method ERGAN performs better than any non-deep-learning method, but worse than the deep-learning methods with word embedding, i.e., DM, DTAL and ERGAN+WE. Nonetheless, our method ERGAN+WE outperforms all the baseline, including two deep-learning methods DM and DTAL, over all databases they have the results.

Observation 1. *With sufficient training data,* ERGAN+WE *performs better than* ERGAN *due to the power of word embedding for records.* ERGAN+WE *has superior performance against all the baselines consistently.*

Performance comparison under 20% training set. To understand how performance may change when the amount of training data is reduced, we conduct an experiment on the supervised methods by splitting the datasets into 20% for training and 80% for testing. DTAL is excluded in this experiment because the original paper does not have results in this setting and also no code is available.

Fig. 7.3 shows the results of precision, recall and f-measure over four datasets. LR performs the worst except for precision on DBLP-ACM. The performance of the semi-supervised method SBC is better than LR and SVM, and comparable with XGBoost w.r.t. f-measure. This is because SBC takes the advantage of label propagation but its classifier Adaboost is not as powerful as XGBoost. For the deep-learning methods, the performance of DM drops signif-

icantly on DBLP-ACM and DBLP-Scholar and even worse than ERGAN which is different from the case of 60% training in Table 7.1. ERGAN+WE still performs the best on all datasets w.r.t. all the three measures. Overall, compared with Table 7.1, the performance of SVM, LR and DM is affected considerably on one or more datasets, whereas SBC, XGBoost, ERGAN and ERGAN+WE remain comparable performance.

Observation 2. *With reduced but still considerable training data, the performance of* ERGAN *and* ERGAN+WE *remains strong and consistent. This is because* ERGAN *and* ERGAN+WE *benefit from its adversarial learning architecture and diversity and propagation modules.*

Performance comparison under less than 10% training set. To study performance under a limited number of samples with real labels, we further conduct an experiment using only a small percentage for training, ranging from 0.1% to 10% of the datasets, and the rest for testing.

Fig. 7.4 shows the experimental results. ERGAN performs best among all the methods over all the datasets when training data is below 1%. ERGAN+WE performs poorly in this range. However, the performance of ERGAN+WE increases rapidly with increasing training data and exceeds all the other methods on all the datasets when training data reaches 10%. LR and DM have a similar trend as ERGAN+WE, but perform significantly worse. The semi-supervised method SBC performs better than ERGAN+WE only when training data is small, i.e., below 0.2% for Cora and below 0.9% for DBLP-ACM, DBLP-Scholar and NCVoter. The performance of SVM and XGBoost varies in datasets, i.e., perform well on Cora and DBLP-ACM, but badly on DBLP-Schloar and NCVoter. This demonstrates that the performance of SVM and XGBoost is sensitive to the imbalance rate of a dataset, and they fail to handle imbalanced data when no sufficient training data is available. For NCVoter, due to a clear boundary existing between matches and non-matches in the underlying distribution, the performance of all the methods that perform poorly for small training data can be dramatically improved after using 0.6% or more training data. In general, we may conclude that, compared with the cases of 60% and 20% training in Table 7.1 and Fig. 7.3, the performance gain of the methods with word embedding against the methods without word embedding does not exist anymore. Instead, the methods with word embedding performs worse than most of the methods without word embedding when training data is small, i.e., below 1%.

Observation 3. *When decreasing training data,* ERGAN+WE *gradually performs worse than* ERGAN+WE. *This is because,* ERGAN+WE *transforms samples into a high dimensional space through word embedding and thus requires much more labels in training than* ERGAN.

7.5.2.2 Ablation Analysis

We conduct an ablation study to evaluate the effects of the key components of ERGAN, including the adversarial learning architecture, the diversity module and the propagation module, under different label costs, ranging from 0.1% to 60% for training. The results are presented in Table 7.2. We observe that the performance of all the methods ERNN, ERGAN-D, ERGAN-P and ERGAN become stable and gradually converge when the label cost increases, e.g. in the case of 60% training. Nonetheless, ERGAN performs the best among all the methods, and the

Table 7.2: Comparison of f-measure results with 0.1%, 1%, 20% and 60% training for ablation analysis.

Datasets	Cora				DBLP-ACM			
	0.1%	1%	20%	60%	0.1%	1%	20%	60%
ERNN	84.46	90.67	91.43	92.78	88.05	95.68	98.20	98.22
ERGAN-D	79.87	85.14	91.27	92.97	0	93.30	97.16	98.21
ERGAN-P	85.18	90.76	91.42	93.03	92.67	95.96	98.21	98.23
ERGAN	87.45	91.07	91.54	93.03	96.89	96.93	98.22	98.23
Datasets	**DBLP-Scholar**				**NCVoter**			
	0.1%	1%	20%	60%	0.1%	1%	20%	60%
ERNN	82.76	83.17	86.71	87.73	99.39	100	100	100
ERGAN-D	0	78.85	83.43	88.29	0	99.58	100	100
ERGAN-P	83.43	85.34	86.55	88.32	99.39	99.79	100	100
ERGAN	84.23	85.85	86.86	88.32	99.45	100	100	100

performance of the other methods varies in different datasets. In the following, we will discuss how each key component of ERGAN may affect the performance.

Adversarial learning architecture. The performance of ERNN generally lies in between ERGAN-D and ERGAN-P, and significantly worse than ERGAN. This indicates that the use of adversarial learning architecture by ERGAN helps to improve the performance, particularly when training data is limited, e.g., for 0.1% training, ERGAN improves around 3% on Cora and more than 8% on DBLP-ACM upon ERNN.

Diversity module. In Table 7.2, the results of ERGAN-D are the worst among all the methods over all the datasets. This indicates that diverse samples are more informative for model training, which can improve the label efficiency. Specifically, with 0.1% training, ERGAN-D fails to work (i.e., f-measure value is 0) on three datasets except for Cora. This is because ERGAN-D lacks the diversity module and can only randomly select samples for training. As a result, all training samples are selected from the majority class (non-matches), and accordingly no matched sample can be classified correctly by ERGAN-D, i.e., all the samples are classified as non-matches. Since datasets in entity resolution applications are usually highly imbalanced, training data without diversity may hardly contain samples from the minority class (matches) when labels are limited, thus leading to poor performance.

Propagation module. Table 7.2 shows that ERGAN-P generally has better performance than ERNN and ERGAN-D, and thus it may affect the performance of ERGAN least compared with the other two key components: the adversarial learning architecture and the diversity module, especially when the label cost is small, e.g. 0.1% and 1% training. Additionally, when the label cost is 60%, the performance of ERGAN-P and ERGAN is the same. This is because samples with real labels in 60% training data can provide sufficient information for learning, and the propagation of samples with pseudo labels becomes unnecessary.

Observation 4. *In* ERGAN*, all the three key components, i.e., the adversarial learning architecture, the diversity module and the propagation module, are necessary, each serving as an integral part of the entire framework.*

Table 7.3: Comparison of f-measure results under extremely limited real-labeled samples. The methods SVM, LR, XGBoost, DM, ERGAN-D, and ERGAN+WE have the f-measure value 0 in all these settings and are thus excluded from the table.

Dataset	Label Cost (#samples)	Methods			
		SBC	ERNN	ERGAN-P	ERGAN
Cora	50	0	0.6648	0.7358	**0.7735**
	100	0	0.7684	0.7960	**0.8083**
	200	0.303	0.7742	0.8156	**0.8314**
	500	0.7629	0.8234	0.8493	**0.8691**
DBLP-ACM	50	0	0.6694	0.8492	**0.8869**
	100	0	0.7261	0.9143	**0.9673**
	200	0	0.7463	0.9151	**0.9656**
	500	0	0.8013	0.9174	**0.9681**
DBLP-Scholar	50	0	0.0043	0.6777	**0.7760**
	100	0	0.0536	0.7335	**0.8045**
	200	0	0.6869	0.7869	**0.8124**
	500	0	0.7903	0.8256	**0.8372**
NCVoter	50	0	0.3603	0.6389	**0.7192**
	100	0	0.8202	0.9091	**0.9532**
	200	0	0.9289	0.9431	**0.9583**
	500	0	0.9724	**0.9740**	**0.9740**

7.5.2.3 Extremeness Test

In the previous experiments, ERGAN has demonstrated strong and consistent performance when training data is reduced from 60% to 0.1%. However, a question left is: what are the minimum label costs required by ERGAN to achieve reasonably performance? To answer this, we conduct an experiment under extremely limited label cost, ranging from 50 to 500 samples with real labels. Table 7.3 shows the results of our experiment. Note that we exclude the results of the methods SVM, LR, XGBoost, DM, ERNN, ERGAN-D and ERGAN+WE from this table because they fail to work when the label costs are below 500, i.e., f-measure value is 0 in all the settings in Table 7.3.

In Table 7.3, the results of SBC are almost 0 except for the cases when the label cost is 200 and 500 on Cora. This shows that SBC as a semi-supervised method can perform better than other fully supervised methods SVM, LR, XGBoost and DM under extremely limited label cost. Moreover, the performance of SBC is affected by the imbalance rate, and SBC fails to perform when the imbalance rate of a dataset is high.

We also notice that ERGAN can perform reasonably well on all the datasets even with 50 labels, and achieve good performance using only 500 labels. Moreover, ERNN, ERGAN-P and ERGAN have results in all the setting, but ERGAN-D does not have results in any of these settings. This is due to the diversity module, which can effectively select balanced training data to maximize the use of labels under extremely limited label cost. ERGAN-P performs better

than ERNN on all the datasets. It indicates that performance may be harmed rather than helped by propagation if the quality of pseudo labels being propagated is not guaranteed. It is worthy to note that, for the cases of 50 and 100 labels on DBLP-Scholar dataset, we find that the reason why ERNN has very low feature values is because of high recall values (i.e., 0.749 for 50 labels and 0.777 for 100 labels) but low precision values (i.e., 0.002 for 50 labels and 0.028 for 100 labels). It means ERNN incorrectly classifies many non-matches as matches (i.e., false positives) and then propagates them into training data, leading to poor performance. With the increase in the label cost, this phenomenon is alleviated.

Observation 5. ERGAN *can achieve good performance even with extremely limited labels. This is because the label generator G and the discriminator D are trained adversarially in* ERGAN *such that G uses the diversity module to balance the selection of samples from different classes and D propagates samples with high-quality psuedo labels into training.*

7.6 Summary

In this chapter, we have proposed a novel method, called ERGAN, to solve the entity resolution classification problem with very limited labeled samples. ERGAN incorporates the diversity of samples into sampling, prior to training the models. ERGAN consists of a label generator G to generate pseudo labels for unlabeled samples, and a discriminator D to distinguish samples with pseudo labels from samples with real labels.

Chapter 8

Conclusions and Future Work

In this thesis, we have studied some tasks that are important during the entity resolution process, such as learning blocking schemes and training classifiers. While existing work focuses on how to learn good blocking schemes and how to train a classification model w.r.t. the accuracy and scalability, we have noticed the following challenges in entity resolution: highly imbalanced data distributions and difficulties of obtaining labels in real-life applications. Thus, we have targeted at solving such challenges. Specifically, we have proposed the following approaches:

- In Chapter 4, we have used active learning techniques to develop a blocking scheme learning approach. Our approach overcomes the weaknesses of the existing work in two aspects: (1) Previously, supervised blocking scheme learning approaches require a large number of labels for learning a blocking scheme, which is an expensive task for entity resolution; (2) Existing unsupervised blocking scheme learning approaches generate training sets based on the similarity of record pairs, instead of their true labels, thus the training quality can not be guaranteed. Our experimental results show that our proposed approach outperforms the baseline approaches under a pre-defined error rate within a label budget.

- In Chapter 5, we have proposed a scheme skyline learning framework called skyblocking, which integrates skyline query techniques and active learning techniques into learning a set of optimal blocking schemes under different constraints and a limited label budget. We have tackled the class imbalance problem by solving the balanced sampling problem. We have also proposed the scheme extension strategy to reduce the searching space and label costs. We have further developed three algorithms for efficiently learning scheme skylines.

- In Chapter 6, we have proposed a novel learning-based active learning framework called Learning-To-Sample. We use the state-of-the-art machine learning models to solve the classification tasks in entity resolution. This framework is composed of a sampling model G and a boosting model F. The boosting model contains a dynamic training set with an increasing number of samples in different iterations. These additional samples are selected iteratively by the sampling model which can learn from the performance of the boosting model through a unified process for two sampling strategies: uncertainty sampling (US) and diversity sampling (DS). The experimental results show that

our approach outperforms all the baselines, particularly when the number of samples is relatively small. In addition to this, our framework can handle the cold start problem and the class imbalance problem.

- In Chapter 7, we have proposed ERGAN, to solve the entity resolution classification problem under a limited number of labeled samples. ERGAN takes the advantages of semi-supervised learning, and it incorporates the diversity of samples into sampling, prior to training the models. This approach consists two key components: (1) with a diversity module, the label generator G is used to generate pseudo labels for unlabeled samples; (2) a discriminator D is used to distinguish samples with pseudo labels from samples with real labels, and a propagation module is designed to propagate high ranking pseudo labeled samples into real labeled sample set. This approach can be extended with word embedding for handling attribute values, leading to an enhanced method, called ERGAN+WE. We have formally proven that even with a limited number of samples with real labels, D and G in ERGAN can converge to the true distribution of samples and their labels. Our experimental results show that the performance of our methods beats all the baselines.

Inspired by the success for deep learning models in computer vision [60; 59] and natural language processing [39], entity resolution with deep learning techniques has been widely developed. Compared with traditional entity resolution approaches targeting at solving structured data, they have shown the advantages on solving unstructured data, i.e., no attribute is specified in the dataset, which can achieve much better performance [113]. The most recent work using Graph Convolutional Network (GCN) considers entity resolution in a token-centric manner rather than attribute-centric [99].

Apart from graph-based neural networks, language models are also trained and applied for non-structured data, i.e., sentences, paragraphs and documents w.r.t. natural language. some work has been published for entity resolution based on the pre-trained language models and achieved quite comparable results [159]. Due to the challenge that the labels for training samples are hard to achieve, transfer learning technique has also been adopted in entity resolution, which helps to reduce the label cost using pre-trained models [81]. Further researches are still necessary on deep learning-based entity resolution classification. For example, integrating active learning techniques into deep learning models, which will be useful to reduce the label cost in training a model. Active learning techniques can also be used with transfer learning using pre-trained language models. Specific active learning strategies can be defined as heuristics based on the token level record representations. Furthermore, graph-based models can also be developed for classification with active learning techniques.

In the future, we can also further explore how to design an efficient deep learning-based approach for blocking. Skyline querying is a useful technique for decision making. Skyblocking helps the user to select the most suitable blocking schemes in practice. However, at its core, the efficiency and accuracy of the scheme skyline are determined by blocking scheme learning algorithms. Our skyblocking algorithm can be improved in the future with applying some state-of-the-art skyline query algorithms and skyline learning approaches [80; 155] into blocking scheme learning process, thus the performance of skyblocking can also be improved.

Furthermore, we may consider an extended active learning framework which will not be limited to the blocking scheme learning, but also be applied for schema selection and so on.

Bibliography

1. A. Aizawa and K. Oyama. A fast linkage detection scheme for multi-source information integration. In *Web Information Retrieval and Integration, 2005. WIRI'05. Proceedings. International Workshop on Challenges in*, pages 30–39. IEEE, 2005. 23

2. Y. Altowim, D. V. Kalashnikov, and S. Mehrotra. Progressive approach to relational entity resolution. *Proceedings of the VLDB Endowment*, 7(11):999–1010, 2014. 3

3. A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *SIGMOD*, pages 783–794. ACM, 2010. 38

4. A. Arasu, M. Götz, and S. Kaushik. Active learning of record matching packages, July 14 2015. US Patent 9,081,817. 7

5. M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, pages 214–223, 2017. 34

6. S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *International Conference on Machine Learning (ICML)*, 2002. 7

7. R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage; erschienen in: Proceedings of the workshop on data cleaning, record linkage and object consolidation at the ninth acm sigkdd international conference on knowledge discovery and data mining; washington dc; 2003; o. 21

8. K. Bellare, S. Iyengar, A. G. Parameswaran, and V. Rastogi. Active sampling for entity matching. In *SIGKDD*, pages 1131–1139. ACM, 2012. 38

9. Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155, 2003. 8

10. I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *SIAM International Conference on Data Mining*, pages 47–58, 2006. 27

11. M. Bilenko, B. Kamath, and R. J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 87–96. IEEE, 2006. 3, 5, 6, 22, 36, 37, 42, 60

12. P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. 95

13. S. Borzsony, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430. IEEE, 2001. 30

14. L. Breiman. *Classification and regression trees*. Wadsworth International Group, 1984. 33, 79

15. L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 33

16. K. Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003. 31

17. Y. Cao, Z. Chen, J. Zhu, P. Yue, C.-Y. Lin, and Y. Yu. Leveraging unlabeled data to scale blocking for record linkage. In *IJCAI*, volume 22, page 2211, 2011. 5, 6, 22, 37

18. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. 31

19. T. Chen and C. Guestrin. Xgboost: a scalable tree boosting system. In *international conference on Knowledge Discovery and Data mining (SIGKDD)*, 2016. 27, 33, 79, 80, 95, 97

20. S. Chester and I. Assent. Explanations for skyline query results. In *EDBT*, pages 349–360, 2015. 49

21. J. Chomicki, P. Ciaccia, and N. Meneghetti. Skyline queries, front and back. *SIGMOD Record*, 42(3):6–18, 2013. 29, 49

22. J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In *ICDE*, pages 717–719. IEEE, 2003. 30

23. J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting: Theory and optimizations. In *IIPWM*, pages 595–604. Springer, 2005. 30

24. P. Christen. Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *international conference on Knowledge Discovery and Data mining (SIGKDD)*, pages 151–159. ACM, 2008. 7, 27, 95, 97

25. P. Christen. Development and user experiences of an open source data cleaning, deduplication and record linkage system. *ACM SIGKDD Explorations Newsletter*, 11(1):39–48, 2009. 26, 27

26. P. Christen. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media, 2012. 1, 2, 3, 19, 20, 21, 26, 31

27. P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *TKDE*, 24(9):1537–1555, 2012. 2, 6, 14, 21, 44, 60

28. P. Christen et al. Towards parameter-free blocking for scalable record linkage. 2007. 21

29. P. Christen, D. Vatsalan, and Q. Wang. Efficient entity resolution with adaptive and interactive training data selection. In *ICDM*, pages 727–732. IEEE, 2015. 49

30. V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, and K. Stefanidis. End-to-end entity resolution for big data: A survey. *arXiv preprint:1905.06397*, 2019. 28, 29

31. H.-M. Chu and H.-T. Lin. Can active learning experience be transferred? In *Proceedings of the 16th International Conference on Data Mining (ICDM)*, 2016. 32

32. T. Churches, P. Christen, K. Lim, and J. X. Zhu. Preparation of name and address data for record linkage using hidden markov models. *BMC Medical Informatics and Decision Making*, 2(1):9, 2002. 20

33. D. Clark. Practical introduction to record linkage for injury research. *Injury Prevention*, 10(3):186–191, 2004. 19

34. W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, pages 73–78, 2003. 26

35. W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 475–480. ACM, 2002. 5, 23

36. A. Culotta and A. McCallum. Reducing labeling effort for structured prediction tasks. In *Proceedings of the AAAI conference on artificial intelligence*, 2005. 31

37. S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on machine learning*, pages 208–215. ACM, 2008. 30, 36

38. Y. Deng, K. Chen, Y. Shen, and H. Jin. Adversarial active learning for sequences labeling and generation. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, 2018. 34, 69, 70, 78

39. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 28, 29, 104

40. P. Donmez and J. G. Carbonell. Paired-sampling in density-sensitive active learning. 2008. 69

41. A. V. Dorogush, V. Ershov, and A. Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018. 33

42. O. Dovrat, I. Lang, and S. Avidan. Learning to sample. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 32

43. U. Draisbach and F. Naumann. A comparison and generalization of blocking and windowing algorithms for duplicate detection. In *International Workshop on QDB*, pages 51–56, 2009. 5

44. M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467, 2018. 19, 28

45. S. Ertekin, J. Huang, L. Bottou, and L. Giles. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the international Conference on Information and Knowledge Management (CIKM)*, 2007. 7, 31, 69

46. W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about record matching rules. *Proceedings of the VLDB Endowment*, 2(1):407–418, 2009. 27

47. I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969. 1, 21, 22, 44, 60

48. Z. Ferdowsi, R. Ghani, and R. Settimi. Online active learning with imbalanced classes. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1043–1048. IEEE, 2013. 31

49. J. Fisher, P. Christen, and Q. Wang. Active learning based entity resolution using markov logic. In *PAKDD*, pages 338–349. Springer, 2016. 7, 31, 36, 49

50. J. Fisher, P. Christen, Q. Wang, and E. Rahm. A clustering-based framework to control block sizes for entity resolution. In *SIGKDD*, pages 279–288. ACM, 2015. 1, 5, 21, 23

51. M. Fortini, B. Liseo, A. Nuccitelli, and M. Scanu. On bayesian record linkage. *Research in Official Statistics*, 4(1):185–198, 2001. 26

52. Y. Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 1995. 27, 33

53. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 1997. 77

54. Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1996. 33

55. J. Friedman, T. Hastie, R. Tibshirani, et al. Additive logistic regression: a statistical view of boosting. *The annals of statistics*, 2000. 33

56. C. Fu, X. Han, L. Sun, B. Chen, W. Zhang, S. Wu, and H. Kong. End-to-end multi-perspective matching for entity resolution. In *IJCAI*, pages 4961–4967, 2019. 28

57. C. W. Gardiner et al. *Handbook of stochastic methods*, volume 3. springer Berlin, 1985. 93

58. L. Getoor and A. Machanavajjhala. Entity resolution for big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1527–1527. ACM, 2013. 3

59. I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016. 28, 34, 87, 104

60. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems (NeurIPS)*, pages 2672–2680, 2014. 28, 34, 87, 93, 104

61. Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems (NeurIPS)*, pages 529–536, 2005. 94

62. A. Gruenheid, X. L. Dong, and D. Srivastava. Incremental record linkage. *VLDB Endowment*, 7(9):697–708, 2014. 2, 3

63. J. Hammersley. *Monte carlo methods*. Springer Science & Business Media, 2013. 32

64. J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011. 27

65. M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *Sigmod Record*, volume 24, pages 127–138. ACM, 1995. 21, 22

66. T. N. Herzog, F. J. Scheuren, and W. E. Winkler. *Data quality and record linkage techniques*. Springer Science & Business Media, 2007. 26

67. T. K. Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995. 33

68. A. Holub, P. Perona, and M. C. Burl. Entropy-based active learning for object recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, 2008. 31, 69

69. W.-N. Hsu and H.-T. Lin. Active learning by learning. In *Proceedings of the Twenty-Ninth AAAI conference on artificial intelligence*, 2015. 31, 69

70. Y. Hu, Q. Wang, D. Vatsalan, and P. Christen. Regression classifier for improved temporal record linkage. 2016. 2

71. A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010. 7

72. P. Jain and A. Kapoor. Active learning for large multi-class problems. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 7, 31

73. P. Jamshidi, M. Velez, C. Kästner, and N. Siegmund. Learning to sample: Exploiting similarities across environments to learn performance models for configurable systems. In *Proceedings of the 26th Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2018. 32

74. M. A. Jaro. Probabilistic linkage of large public health data files. *Statistics in medicine*, 14(5-7):491–498, 1995. 19

75. G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279. ACM, 2003. 19

76. L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, and A. Hauptmann. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2078–2086, 2014. 31, 75, 78

77. L. Jin, C. Li, and S. Mehrotra. Efficient record linkage in large data sets. In *Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings. Eighth International Conference on*, pages 137–146. IEEE, 2003. 23

78. P. Jokinen, J. Tarhio, and E. Ukkonen. A comparison of approximate string matching algorithms. *Software: Practice and Experience*, 26(12):1439–1458, 1996. 24

79. A. Jurek, J. Hong, Y. Chi, and W. Liu. A novel ensemble learning approach to unsupervised record linkage. *Information Systems*, 71:40–54, 2017. 27

80. C. Kalyvas and T. Tzouramanis. A survey of skyline query processing. *arXiv preprint arXiv:1704.01788*, 2017. 29, 49, 104

81. J. Kasai, K. Qian, S. Gurajada, Y. Li, and L. Popa. Low-resource deep entity resolution with transfer and active learning. *arXiv preprint arXiv:1906.08042*, 2019. 28, 95, 97, 104

82. G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017. 33

83. M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 1994. 33

84. M. Kejriwal and D. P. Miranker. An unsupervised algorithm for learning blocking schemes. In *ICDM*, pages 340–349. IEEE, 2013. 3, 5, 6, 22, 35, 36, 37, 42, 44, 60

85. M. Kejriwal and D. P. Miranker. A two-step blocking scheme learner for scalable link discovery. In *OM*, pages 49–60, 2014. 5, 22

86. M. Kejriwal and D. P. Miranker. A dnf blocking scheme learner for heterogeneous datasets. *arXiv preprint arXiv:1501.01694*, 2015. 6, 22

87. M. Kejriwal and D. P. Miranker. Semi-supervised instance matching using boosted classifiers. In *European Semantic Web Conference*, pages 388–402. Springer, 2015. 7, 27, 95, 97

88. I. Keles and K. Hose. Skyline queries over knowledge graphs. In *International Semantic Web Conference*, pages 293–310. Springer, 2019. 29

89. H. Keskustalo, A. Pirkola, K. Visala, E. Leppänen, and K. Järvelin. Non-adjacent digrams improve matching of cross-lingual spelling variants. In *String Processing and Information Retrieval*, pages 252–265. Springer, 2003. 21, 23

90. S. Kim, Y. Song, K. Kim, J.-W. Cha, and G. G. Lee. Mmr-based active machine learning for bio named entity recognition. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, 2006. 7, 31

91. A. Kisielewicz. A solution of dedekincts problem on the number of isotone boolean functions. *J. reine angew. math*, 386:139–144, 1988. 42

92. P. Konda, S. Das, A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton, and S. Prasad. Magellan: toward building entity matching management systems over data science stacks. *Proceedings of the VLDB Endowment*, 9(13):1581–1584, 2016. 27, 95, 97

93. K. Konyushkova, R. Sznitman, and P. Fua. Learning active learning from data. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 2017. 31, 32, 69, 70, 78

94. H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197–210, 2010. 51

95. H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *VLDB Endowment*, 3(1-2):484–493, 2010. 13, 59

96. N. Krieger. Social class and the black/white crossover in the age-specific incidence of breast cancer: a study linking census-derived data to population-based registry records. *American Journal of Epidemiology*, 131(5):804–814, 1990. 19

97. K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):377–439, 1992. 23

98. D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the conference on Information Retrieval (SIGIR)*, 1994. 31, 69

99. B. Li, W. Wang, Y. Sun, L. Zhang, M. A. Ali, and Y. Wang. Grapher: Token-centric entity resolution with graph convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8172–8179, 2020. 29, 104

100. C. Li, J. Li, G. Wang, and L. Carin. Learning to sample with adversarially learned likelihood-ratio. 2018. 32

101. X. Li, Y. Wu, M. Ester, B. Kao, X. Wang, and Y. Zheng. Semi-supervised clustering in attributed heterogeneous information networks. In *International Conference on World Wide Web (WWW)*, pages 1621–1629, 2017. 7

102. Y. Li, J. Li, Y. Suhara, A. Doan, and W.-C. Tan. Deep entity matching with pre-trained language models. *arXiv preprint arXiv:2004.00584*, 2020. 29

103. X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: The k most representative skyline operator. In *ICDM*, pages 86–95. IEEE, 2007. 29

104. V. S. Lokhande, S. Wang, M. Singh, and J. Yarkony. Accelerating column generation via flexible dual optimal inequalities with application to entity resolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1593–1602, 2020. 28

105. C. D. Manning, H. Schütze, et al. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999. 14

106. L. Maystre and M. Grossglauser. Just sort it! a simple and effective approach to active preference learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017. 69

107. A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. Citeseer, 2000. 21

108. M. Michelson and C. A. Knoblock. Learning blocking schemes for record linkage. In *AAAI*, pages 440–445, 2006. 5, 6, 22, 35

109. M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint:1411.1784*, 2014. 34

110. T. M. Mitchell et al. Machine learning. wcb, 1997. 22

111. A. E. Monge, C. Elkan, et al. The field matching problem: Algorithms and applications. In *KDD*, pages 267–270, 1996. 26

112. M. Morse, J. M. Patel, and H. V. Jagadish. Efficient skyline computation over low-cardinality domains. In *VLDB*, pages 267–278. VLDB Endowment, 2007. 30

113. S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34. ACM, 2018. 8, 19, 28, 95, 97, 104

114. M. Murugesan, W. Jiang, C. Clifton, L. Si, and J. Vaidya. Efficient privacy-preserving similar document detection. *The VLDB Journal—The International Journal on Very Large Data Bases*, 19(4):457–475, 2010. 19

115. G. Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001. 24, 25

116. H. Nie, X. Han, B. He, L. Sun, B. Chen, W. Zhang, S. Wu, and H. Kong. Deep sequence-to-sequence entity matching for heterogeneous entity resolution. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 629–638, 2019. 28

117. M. Odell and R. Russell. The soundex coding system. *US Patents*, 1261167, 1918. 21

118. K. O'Hare, A. Jurek, and C. de Campos. A new technique of selecting an optimal blocking method for better record linkage. *Information Systems*, 77:151–166, 2018. 52

119. G. Papadakis, E. Ioannou, C. Niederée, and P. Fankhauser. Efficient entity resolution for large heterogeneous information spaces. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 535–544. ACM, 2011. 21

120. G. Papadakis, E. Ioannou, T. Palpanas, C. Niederee, and W. Nejdl. A blocking framework for entity resolution in highly heterogeneous information spaces. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2665–2682, 2012. 21

121. G. Papadakis, G. Koutrika, T. Palpanas, and W. Nejdl. Meta-blocking: Taking entity resolutionto the next level. *TKDE*, 26(8):1946–1960, 2014. 5, 21, 23

122. G. Papadakis, G. Papastefanatos, and G. Koutrika. Supervised meta-blocking. *VLDB Endowment*, 7(14):1929–1940, 2014. 5, 21, 23

123. D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, pages 467–478. ACM, 2003. 30

124. R. Popp and J. Poindexter. Countering terrorism through information and privacy protection technologies. *IEEE Security & Privacy*, 4(6), 2006. 19

125. D. Pyle. *Data preparation for data mining*. morgan kaufmann, 1999. 7

126. B. Qian, X. Wang, N. Cao, H. Li, and Y.-G. Jiang. A relative similarity based method for interactive patient risk prediction. *Data Mining and Knowledge Discovery*, 2015. 7, 31

127. V. Raghavan, P. Bollmann, and G. S. Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems (TOIS)*, 7(3):205–229, 1989. 14

128. E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000. 20

129. G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001. 27, 95

130. T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in neural information processing systems (NeurIPS)*, pages 2234–2242, 2016. 34

131. A. D. Sarma, A. Lall, D. Nanongkai, R. J. Lipton, and J. Xu. Representative skylines using threshold-based preference distributions. In *ICDE*, pages 387–398. IEEE, 2011. 29, 49

132. G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *ICML*, volume 2, page 6. Citeseer, 2000. 7, 69

133. B. Settles. Active learning literature survey. 2010. 7, 30, 31, 69

134. B. Settles. Active learning. *Synthesis Lectures on AIML*, 6(1):1–114, 2012. 30

135. J. Shao and W. Qing. Active blocking scheme learning for entity resolution. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2018. 5, 49, 79

136. J. Shao, Q. Wang, and F. Liu. Learning to sample: an active learning framework. In *International Conference on Data Mining (ICDM)*, 2019. 7, 92

137. R. Singh, V. V. Meduri, A. Elmagarmid, S. Madden, P. Papotti, J.-A. Quiané-Ruiz, A. Solar-Lezama, and N. Tang. Synthesizing entity matching rules by examples. *VLDB Endowment*, 11(2):189–202, 2017. 27

138. J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint:1511.06390*, 2015. 34, 89

139. R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998. 73

140. V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003. 33

141. Y. Tao, L. Ding, X. Lin, and J. Pei. Distance-based representative skyline. In *ICDE*, pages 892–903. IEEE, 2009. 29

142. K.-S. Teong, L.-K. Soon, and T. T. Su. Schema-agnostic entity matching using pre-trained language models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2241–2244, 2020. 29

143. I. Triguero, S. García, and F. Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42(2):245–284, 2015. 94

144. B. Van Berkel and K. De Smedt. Triphone analysis: a combined method for the correction of orthographical and typographical errors. In *Proceedings of the second conference on Applied natural language processing*, pages 77–83. Association for Computational Linguistics, 1988. 23

145. V. S. Verykios, G. V. Moustakides, and M. G. Elfeky. A bayesian decision model for cost optimal record matching. *The VLDB Journal*, 12(1):28–40, 2003. 26

146. Q. Wang, M. Cui, and H. Liang. Semantic-aware blocking for entity resolution. *TKDE*, 28(1):166–180, 2016. 5, 35

147. Q. Wang, J. Gao, and P. Christen. A clustering-based framework for incrementally repairing entity resolution. In *PAKDD*, pages 283–295. Springer, 2016. 2, 27, 28, 35

148. Q. Wang, K.-D. Schewe, W. Wang, et al. Provenance-aware entity resolution: Leveraging provenance to improve quality. In *DASFAA (1)*, pages 474–490, 2015. 27

149. Q. Wang, D. Vatsalan, and P. Christen. Efficient interactive training selection for large-scale entity resolution. In *PAKDD*, pages 562–573. Springer, 2015. 3, 7, 37, 49

150. S. E. Whang and H. Garcia-Molina. Entity resolution with evolving rules. 2010. 27

151. S. E. Whang, D. Marmaros, and H. Garcia-Molina. Pay-as-you-go entity resolution. *IEEE Transactions on Knowledge and Data Engineering*, 25(5):1111–1124, 2013. 19

152. S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. Entity resolution with iterative blocking. In *SIGMOD*, pages 219–232. ACM, 2009. 23

153. I. H. Witten, A. Moffat, and T. C. Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999. 23

154. R. Wu, S. Chaba, S. Sawlani, X. Chu, and S. Thirumuruganathan. Zeroer: Entity resolution using zero labeled examples. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1149–1164, 2020. 29

155. M. Xie, R. C.-W. Wong, and A. Lall. An experimental survey of regret minimization query and variants: bridging the best worlds between top-k query and skyline query. *The VLDB Journal*, 29(1):147–175, 2020. 104

156. Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, 2007. 31

157. Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 2015. 7, 31, 32, 69, 78

158. D. Zhang, L. Guo, X. He, J. Shao, S. Wu, and H. T. Shen. A graph-theoretic fusion framework for unsupervised entity resolution. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 713–724. IEEE, 2018. 27, 95, 97

159. C. Zhao and Y. He. Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *The World Wide Web Conference*, pages 2413–2424, 2019. 28, 104