# Active Blocking Scheme Learning For Entity Resolution

Nikhil Acharya[1], Nofel Mahmood[2] and Saurav Ganguli[3]

[1] Informatik III, Universität Bonn, Germany
s6niacha@uni-bonn.de
[2] Informatik III, Universität Bonn, Germany
s6nomaho@uni-bonn.de
[3] Informatik III, Universität Bonn, Germany
s6sagang@uni-bonn.de

**Abstract**

Blocking aims to improve the time efficiency of entity resolution by grouping potentially matched records into the same block. Using blocking scheme learning approach, based on active learning techniques, our approach can learn a blocking scheme to generate high quality blocks within a limited label budget. Active sampling is used to select optimal samples and active branching is used to generate efficient blocking schemes. Our experimental verifications prove this approach outperforms several other baseline approaches. The project is implemented in Scala using the Spark framework.

This task was implemented as part of our course *MA-INF 4223 - Lab Distributed Big Data Analytics*, presented by Dr. Hajira Jabeen and Gezim Sejdiu, Informatik III, Universität Bonn, Germany, during Winter Semester 2018/19.

**Keywords:** Entity Resolution, Blocking Scheme, Active Learning, Active Sampling, Active Branching, Class Imbalance Problem

## 1 Problem Definition

Entity Resolution (ER) is the process of identifying records which represent the same real-word entity from one or more datasets [5]. Conventional ER processes have large time complexities and are expensive as they match each entity with all the other entities in the dataset. Thus, blocking techniques were introduced to improve the time efficiency, by grouping potentially matched records in the same block [4]. This removes the need of having to compare all records, but comparison only occurs with records which are part of the same block. For a given dataset D, the total number of record pairs to be compared is $\frac{|D|*(|D|-1)}{2}$. With blocking, the number of record pairs to be compared is no more than $\frac{m*(m-1)}{2} * \mid B \mid$, where m is the number of records in the largest block and $\mid B \mid$ is the number of blocks.

The existing blocking schemes have a particular problem due to the high imbalance of match and non-match labels, called the class imbalance problem. Additionally, it is difficult to obtain a blocking quality using unsupervised learning. The active blocking scheme learning described in the research paper [1] incorporates active learning techniques into the blocking scheme learning process. This approach actively learns the blocking scheme based on a set of blocking predicates, which efficiently solves the class imbalance problem. The results indicate that this approach gives high quality blocks within a specified error bound and a limited label budget.

## 2 Approach

The basic approach used in solving the problem is inspired by the research paper [1]. It is divided into two main stages: *active sampling* and *active branching*. These two combine to give us our algorithm *Active Scheme Learning* which is described in Section 3.

Our algorithm includes the labeling of samples semi-automatically, which is performed by the human oracle function. In the end we would like to deduce an optimum scheme which can divide maximum number of samples into blocks with high accuracy, which is verified by comparison with human oracle.

The algorithm uses supervised learning technique which needs good number of match and no match samples. So we have used a dataset with sufficient cases of match and no match. The class imbalance ratio of dataset is not too high. and Figure 1 below illustrates the architecture of the proposed approach. We briefly describe the two main tasks.

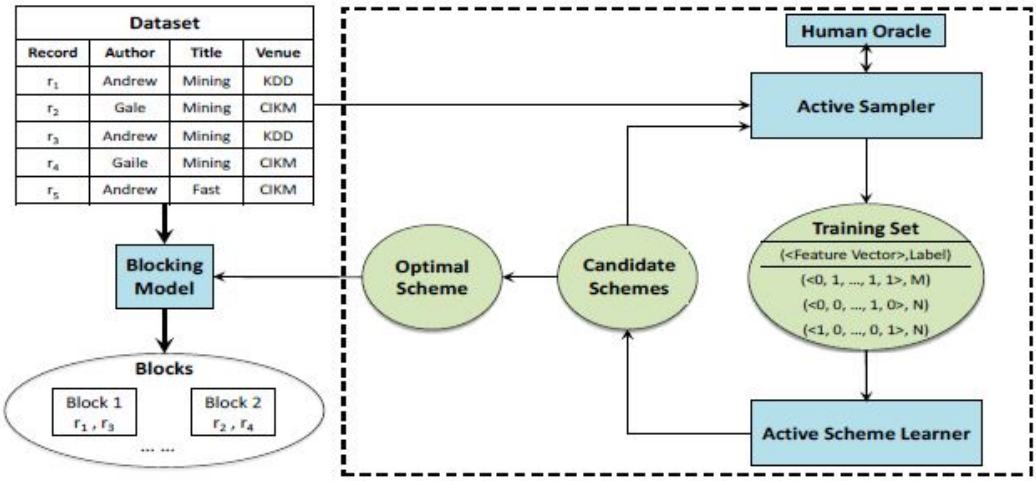


Figure 1: Architecture of the proposed algorithm

### 2.1 Active Sampling

When our algorithm begins with learning the dataset, there are both match and non-match samples, in a way that non-match records marginally outnumber the match records. However, it is useful to have a balanced sample, so that quality blocking schemes can be achieved. Thus, our active sampling strategy is used to overcome the class imbalance problem by selecting informative training samples. For active sampling we need sufficient number of samples to be added to create a balance. We check the balance rate using a gamma function. For a blocking scheme s and a non-empty feature vector set X, the balance rate of X in terms of s, is defined as:

$$\gamma(s, X) = \frac{\mid \{x_i \epsilon X | s(X_i) = true\} \mid - \mid \{x_i \epsilon X | s(X_i) = false\} \mid}{\mid X \mid} \quad (1)$$

### 2.2 Active Branching

In active branching, we assume that human oracle has labelled the balanced samples correctly to generate the training data in a way that we can achieve a good scheme.We can construct our blocking schemes from the balanced training data, in the form of conjunctions or disjunctions. With an initial blocking scheme, it can be extended in further iterations using the two lemmas described in the research paper [1].

**Lemma 1.** For the conjunction of s1 and s2, the following holds:

$$| fp(B_{si}) | \geq | fp(B_{s1 \wedge s2}) | \tag{2}$$

where i = 1, 2

**Lemma 2.** For the disjunction of s1 and s2, the following holds:

$$\frac{| fn(B_{si}) |}{| tp(B_{si}) |} \geq \frac{| fn(B_{s1 \vee s2}) |}{| tp(B_{s1 \vee s2}) |} \tag{3}$$

where i = 1, 2

### 2.3 Data Structures

- Spark Dataframes [2] [3]

### 2.4 Dataset

The research paper mentions the use of four real-world datasets for the experiments. However, these datasets are currently unavailable. Therefore, we used the sample dataset given in the paper and modified it for our experiments.

## 3 Implementation

We use the proposed algorithm in the paper, called Active Scheme Learning, to generate a set of blocking schemes and learn one optimal scheme. For this we use two strategies called Active Sampling and Active Branching.We begin with our dataset as input and provide a pre-defined error rate threshold $\varepsilon \in [0, 1]$, a sample size k and a set of blocking predicates. A blocking predicate is a combination of a blocking function and an attribute. We have used a function *Soundex* for comparing record pairs for each attribute. This is a function in the Spark library [2], that accepts two strings and performs a pairwise string matching. Using this we generate a sample dataset, the rows of which are nothing but feature vectors, which are deduced from the pairwise comparison of each record in the sample. The value 1 represents similar values and 0 represents dissimilar values. A set of random samples, defined by the sample size k, is selected and provided as input to the active sampler.

The initial set of schemes are initialized to the blocking predicates of each attribute. Now we check the balance rate of each scheme using a gamma function. We subtract the count of 0s and 1s and divide them with the total count to get the gamma value, which is in the range [-1, 1]. A value close to 0 indicates a balanced sample set, otherwise is an unbalanced sample set for that particular scheme. This leads to our case of class imbalance problem. To remove the unbalance, we need to check the notation of balance rate and add an equal number of similar or dissimilar samples.

We pass the balanced scheme set now to a human oracle. This is a semi automated function, as a part of supervised learning of the algorithm. The human oracle labels the rows of the sample set by analyzing similar entities. In case of similar record pairs, it outputs the label as M and label N in case of dissimilar pairs. We have experimented with different parameters for the human oracle, in the range of 50% and 100% match of pairs. This leads us to the generation of our training set, which we use to find optimal schemes.

We calculate the error rate for each scheme by comparing observed values (human oracle) and calculated scheme values obtained for each record in the training set. The scheme with the least error rate obtained from the training set is noted as the locally optimal scheme. If the error rate of this scheme is less than $\varepsilon$, then we proceed to evaluate and extend the scheme using Lemma 1. This involves a comparison of false positives of the existing schemes with an extended scheme using conjunction of each values in the candidate scheme. If lemma 1 is satisfied, we add the extended scheme to our set of candidate schemes using conjunction. If the error rate is greater than $\varepsilon$, we proceed with Lemma 2, where we compare the ratio of false negatives and true positives of optimum scheme with the schemes extended by a disjunction from the current set of candidate schemes. If the Lemma 2 is satisfied, the extended scheme is added to the set of candidate schemes using disjunction.

The algorithm runs till the budget limit is exhausted and the optimum schemes obtained by Lemma 1 in each iteration is noted. The optimum scheme of the last iteration is considered to be the most optimum scheme for the training set. The optimum schemes are further evaluated to check the correctness with respect to the dataset.

# 4 Evaluation

Our algorithm determines the optimum candidate scheme based on different variable parameters. By changing these parameters, our results can largely vary. We can change the error rate, budget, sample size to achieve different values for our evaluation techniques. We use the following common measures [6] to evaluate blocking quality:

- *Reduction Ratio* (RR) is one minus the total number of record pairs in blocks divided by the total number of record pairs without blocks, which measures the reduction of compared record pairs.
- *Pairs Completeness* (PC) is the number of true positives divided by the total number of true matches in the dataset.
- *Pairs Quality* (PQ) is the number of true positives divided by the total number of record pairs in blocks.
- *F-measure* (FM) $FM = \frac{2*PC*PQ}{PC+PQ}$ is the harmonic mean of PC and PQ. In addition to these, we define the notion of constraint satisfaction as $CS = \frac{N_s}{N}$, where $N_s$ is the times of learning an optimal blocking scheme by an algorithm and N is the times the algorithm runs.

We have changed sample size ranging from 15 to 60. The sample size should not be too high, since a higher sample size will reduce the scope of learning the algorithm, resulting in less number of iterations. A higher error rate can cause overfitting as it overly learns the training data and may not work on validation data. A lower error rate might result in generating schemes with low blocking quality. For higher budget, we have obtained high reduction ratios. We have obtained high values of pair completeness and pair quality when our budget and sample size is

| Parameter | Value |
|---|---|
| Label Budget | 45 |
| Human Oracle Percentage | 75% |
| Initial Sample Size | 17 |
| Sample size (k) | 54 |
| Error Rate ($\varepsilon$) | 0.4 |
| Reduction Ratio | 0.41176468 |
| Pair Completeness | 0.8666667 |
| Pair Quality | 0.65 |
| F-measure | 0.74285716 |
| Constraint Satisfaction | 1.0 |

Table 1: Optimal Scheme - **Author**

| Parameter | Value |
|---|---|
| Label Budget | 45 |
| Human Oracle Percentage | 75% |
| Initial Sample Size | 17 |
| Sample size (k) | 81 |
| Error Rate ($\varepsilon$) | 0.4 |
| Reduction Ratio | 0.27659577 |
| Pair Completeness | 0.87096775 |
| Pair Quality | 0.7941176 |
| F-measure | 0.8307693 |
| Constraint Satisfaction | 1.0 |

Table 2: Optimal Scheme - **Title & Venue**

| Parameter | Value |
|---|---|
| Label Budget | 120 |
| Human Oracle Percentage | 75% |
| Initial Sample Size | 30 |
| Sample size (k) | 125 |
| Error Rate ($\varepsilon$) | 0.3 |
| Reduction Ratio | 0.7373737 |
| Pair Completeness | 0.67741936 |
| Pair Quality | 0.8076923 |
| F-measure | 0.73684204 |
| Constraint Satisfaction | 1.0 |

Table 3: Optimal Scheme - **Venue & Author**

| Parameter | Value |
|---|---|
| Label Budget | 170 |
| Human Oracle Percentage | 100% |
| Initial Sample Size | 45 |
| Sample size (k) | 163 |
| Error Rate ($\varepsilon$) | 0.3 |
| Reduction Ratio | 0.9415584 |
| Pair Completeness | 1.0 |
| Pair Quality | 1.0 |
| F-measure | 1.0 |
| Constraint Satisfaction | 1.0 |

Table 4: Optimal Scheme - **(Title & Author) & (Review & Author)**

not less. Similarly the F-measure value is also high. Our constraint satisfaction ratio is always 1 or close to 1 because our dataset is not too large and does not have too many columns with large variations. Hence, we obtain the optimal scheme and subsequently the iteration of the algorithm ends because there is no additional data to be evaluated.

For the best evaluation of the algorithm, we have to keep the error rate around 0.4 and label cost around 120 and sample size in between 50 and 120. When the label budget increases, constraint satisfaction also increases. Tables 1-4 below show our results for different input values, wherein we obtained different optimal schemes each time.

# 5 Project Timeline

The project work was uniformly distributed among the three team members over the entire semester. Some tasks required the team effort and discussions, while parts of the code was written individually and then merged to the shared github repository. Table 1 below shows the timeline breakup:

| Task | Start | Finish | Assigned To |
|---|---|---|---|
| Research paper survey | 08.11.2018 | 15.11.2018 | Nikhil, Nofel, Saurav |
| Presentation 1 prep | 16.11.2018 | 22.11.2018 | Nikhil, Nofel, Saurav |
| Dataset sampling | 23.11.2018 | 12.12.2018 | Nikhil, Nofel, Saurav |
| Planning and architecture | 13.12.2018 | 10.01.2019 | Nikhil, Nofel, Saurav |
| Active Sampling | 11.01.2019 | 30.01.2019 | Nofel |
| Scheme generation | 20.01.2019 | 06.02.2019 | Saurav |
| Active branching | 01.02.2019 | 11.02.2019 | Nikhil |
| Result verification | 11.02.2019 | 15.02.2019 | Nikhil, Nofel, Saurav |
| Report | 17.02.2019 | 22.02.2019 | Nikhil, Nofel, Saurav |
| Final presentation prep | 17.02.2019 | 27.02.2019 | Nikhil, Nofel, Saurav |

Table 5: Project Timeline

# 6 References

[1] Jingyu Shao, Qing Wang. *Active Blocking Scheme Learning for Entity Resolution.*
[2] *Apache Spark*, **http://spark.apache.org**
[3] *SANSA-Stack - Scalable Semantic Analytics Stack*, **http://sansa-stack.net/**
[4] Q. Wang, M. Cui, and H. Liang. Semantic-aware blocking for entity resolution. IEEE *Transactions on Knowledge and Data Engineering*, 28(1):166-180, 2016.
[5] Q. Wang, J. Gao, and P. Christen. A clustering-based framework for incrementally repairing entity resolution. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 283-295, 2016.
[6] P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. IEEE *Transactions on Knowledge and Data Engineering*, 24(9):1537-1555, 2012.