Scaling Record Linkage to Non-Uniform Distributed Class Sizes

Steffen Rendle and Lars Schmidt-Thieme

University of Hildesheim, Machine Learning Lab,
Samelsonplatz 1, D-31141 Hildesheim, Germany
{srendle, schmidt-thieme}@ismll.uni-hildesheim.de

Abstract. Record linkage is a central task when information from different sources is integrated. Record linkage models use so-called blockers for reducing the search space by discarding obviously different record pairs. In practice, important problems have Zipf distributed class sizes with some large classes where blocking is not applicable any more. Therefore we propose two novel meta algorithms for scaling arbitrary record linkage models to such data sets. The first one parallelizes problems by creating overlapping subproblems and the second one reduces the search space for large classes effectively. Our evaluation shows that both scaling techniques are effective and are able to scale state-of-the-art models to challenging datasets.

1 Introduction

When data from different sources is collected, objects of different sources may refer to the same underlying entity. For integration of the datasets, duplicates have to be identified. This task is known among others as record linkage [1, 2], duplicate detection [3, 4] and object identification [5]. For example a price comparison system collects offers from different shops that may refer to the same product (see Table 1). Another example are citation strings that refer to the same publication.

Recent models for solving this task rely on machine learning techniques [6, 3, 5, 7]. For scaling with growing problems they use blockers which restrict the pairs that have to be regarded in time-consuming parts. The key problem with these blockers is that they are supposed to return all positive pairs and remove only those pairs that are obviously negative. Although this technique might scale up for some problems with small uniformly distributed class sizes, it cannot be utilized for other distributions of class sizes like Zipf-distribution.

The contributions of this paper are as follows: (i) We show that the class sizes of some important linkage problems are Zipf distributed which leads to $\Omega(n^2/\ln^2 n)$ positive pairs. Thus, these problems cannot be solved with standard blocking techniques. (ii) We provide two novel scaling methods for record linkage that efficiently scale arbitrary linkage models to Zipf distributed data sets.

Table 1. Example of price comparison data

Product Name	Brand	Price	Class Label
Photosmart 435 Digital Camera	Hewlett Packard	118.99	c_1
HP Photosmart 435 16MB memory	HP	110.00	c_1
Canon EOS 300D black Kit 18-55	Canon	786.00	c_2
EOS300D+EF-S18-55	*unspecified*	873.00	c_2
Digital Camera, Olympus, E-300	Olympus	899.00	c_3
Olympus Camedia IR-300 - Digital-Foto	*unspecified*	273.00	c_4

2 Related Work

The problem of record linkage was first formulated by Newcombe [8] and later put into a mathematical model by Fellegi and Sunter [1]. Today state-of-the-art methods use an adaptive approach based on machine learning techniques like classifiers, clustering or markov logic networks [4, 7]. Almost all models for record linkage rely on predicting the equivalence of a pair of objects. As the number of all different pairs is $\frac{n\cdot(n-1)}{2}$ where n is the number of all records, even small problems are not manageable any more. To avoid this problem, record linkage models typically use blockers, which restrict the number of pairs by discarding all obviously different pairs. There have been many proposals for blocking techniques like sorted neighbourhood methods [9], Canopies [10], and adaptive blocking [11, 12]. An overview of blocking techniques is given by Baxter et al. [13].

Blocking works fine if there are lots of classes and class sizes are small. In fact we will show that this does not hold for some important record linkage tasks because they have Zipf distributed class sizes which leads to $\Omega(n^2/\ln^2 n)$ true pairs. This means even a perfect blocker which only returns the true pairs would generate $\Omega(n^2/\ln^2 n)$ pairs. Consequentially no model exclusively relying on blockers can scale to large problems with Zipf distributed class sizes.

There are some studies of record linkage on large datasets [2, 14], but their problems have different characteristics in terms of only two datasets to be merged or very small class sizes. This differs from our problem setting of non-uniformly distributed class sizes that were built up by automatically crawling many sources, like crawling the web. A second main difference is that they use rather simple models for record linkage. The work of Hernández and Stolfo [9] is similar to the above discussed research in terms of small class sizes and simple record linkage models. Similarly to our work, Hernández and Stolfo propose to use clustering for parallelization. They propose hard-clustering in conjunction with their own blocking method of sorted neighborhoods. Whereas inexpensive hard-clustering might be effective for scaling when dealing with small classes, it is difficult to provide high quality splits in problems with large classes. Moreover, parallelizing without any further reduction step of true pairs does not tackle the problem of having $\Omega(n^2/\ln^2 n)$ true pairs.

Reducing the size and complexity of a graph has already been studied in multilevel graph partitioning [15]. In their work a graph $G_0 = (V_0, E_0)$ is iteratively

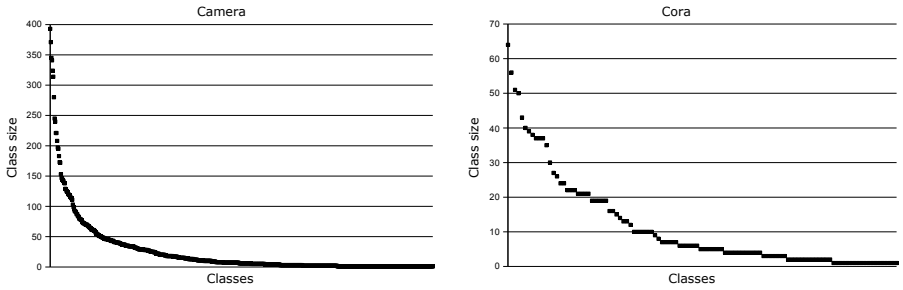

Fig. 1. Distribution of class sizes for the *Camera* and *Cora* dataset.

coarsened to graphs $G_i = (V_i, E_i)$ with $|V_i| > |V_{i+1}|$. Then partitioning is done on the coarsest graph and afterwards the partitioned graph is uncoarsened. Our proposed method for object reduction is related to coarsening as we also reduce the number of objects, perform the expensive calculations on a the small problem and finally expand the solution. Besides this, graph partitioning and record linkage have different problem settings. The differences to record linkage are that in graph partitioning (1) the number of classes is known, (2) all clusters should have roughly equal size and (3) a sparse set of vertices is given in advance.

3 Problem

Figure 1 shows the distribution of class sizes for the bibliographic *Cora* [10] and the *Camera* dataset from a price comparison system[1]. *Cora* contains 1,295 citations to 112 different papers and has 17,184 true pairs. *Camera* has 15,481 offers on 608 digital cameras and has 956,957 true pairs. The classes are sorted by size in descending order. As one can see, the class sizes for both datasets are Zipf-like distributed. Zipf's law[2] states that the most frequent item occurs twice as often as the next frequent one. The third one's frequency is one third of the most frequent class, etc. Applied to class sizes Zipf's law states that the class size of the i-th class is $1/i$ of the size of the largest class, that means the i-th class contains about $\frac{k_{max}}{i}$ objects where k_{max} is the size of the largest class.

The potential reduction rate of all blocking-based scaling techniques depends on the number of true pairs – that means pairs of records referring to the same entity. In Zipf distributed problems, the number of true pairs correlates to the size of the largest class. Thus, we want to estimate the complexity of the largest class k_{max}. Because all class sizes have to sum up to the number of records n,

[1] Mentasys GmbH, Karlsruhe, Germany, http://www.mentasys.de/
[2] For the sake of simplicity, we use an exponent of 1 in all Zipf formulas.

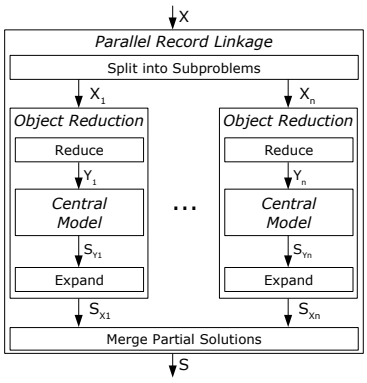

Fig. 2. Combining parallel record linkage with object reduction for scaling a central model.

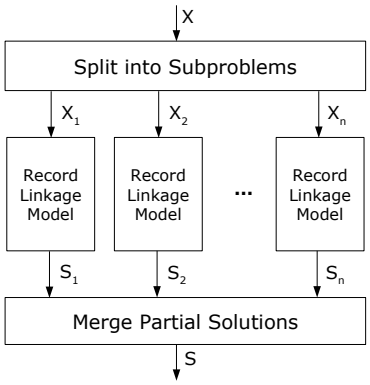

Fig. 3. Parallel Record Linkage: A problem X is split into overlapping subproblems $X_1, \dots X_n$. The subproblems are solved independently in parallel and afterwards are merged to a global solution S.

we can state with Zipf's law:

$$n = \sum_{i=1}^{m} \frac{k_{max}}{i} = k_{max} \cdot \sum_{i=1}^{m} \frac{1}{i} \approx k_{max} \cdot (\ln(m) + \gamma)$$

$$\text{and thus} \quad k_{max} \approx \frac{n}{\ln(m) + \gamma}$$

Where m is the number of all classes, which is unknown in advance and γ is the Euler-Mascheroni constant ($\gamma \approx 0.577$). Now, we can estimate the complexity of k_{max}. As $m \leq n$ also $\ln(m) \leq \ln(n)$ and so we can conclude, that k_{max} grows approximately linear in n. This means, that the size of the largest class k_{max} is in $\Omega(n/\ln n)$. One can conclude that there are $\Omega(n^2/\ln^2 n)$ true pairs in a Zipf distributed record linkage problem.

4 Method

4.1 Scalable Framework

The objective of our framework is to scale up arbitrary record linkage models. In general a record linkage model is a function f_{RL} that generates a partition $f_{RL}(X) \subseteq \mathcal{P}(X)$ of a set of objects X.

We provide two meta algorithms for record linkage that decrease complexity by splitting a problem X in many subproblems $X_1, \dots X_n$ and by reducing the number of objects within a problem X. Both meta models need a record linkage model for solving the modified problems. Basically, our parallelization technique

Algorithm 1 Parallelizing by Canopy-Clustering

```
1: procedure CanopyClustering(X)
   outputs a set P of subproblems for objects X
2:     P ← ∅
3:     C ← X                                   ▷ C is the set of possible centers
4:     while C ≠ ∅ do
5:         x ← random C
6:         Canopy(x) ← {y ∈ X | sim(x, y) > θ_loose}
7:         P ← P ∪ {Canopy(x)}
8:         C ← C \ {y ∈ X | sim(x, y) > θ_tight}
9:     end while
10:    return P
11: end procedure
```

targets problems with many classes whereas our object reduction method targets problems with large classes. Although both algorithms can be used separately, we recommend to combine them so that both aspects are regarded. A useful combination (see figure 2) would be to use parallelizing as outer model, then use object reduction in each parallelized subproblem and solve each reduced subproblem with the record linkage model of your choice – e.g. a classifier based approach like we use in the evaluation chapter.

4.2 Parallel Record Linkage

In general one of the most popular approaches for scaling up systems is parallelizing. Instead of solving one big problem at once, the problem is split in many small problems. These small problems are solved separately and afterwards are merged to a global solution. Our meta model for parallelizing record linkage models work the same way (see figure 3). First, the problem is split into overlapping subproblems. Then each subproblem is solved by another record linkage model and finally the solutions are merged.

Split into Subproblems Parallelizing is a function f_P that generates a partition of overlapping sets, s.t.:

$$\bigcup_{X_i \in f_P(X)} X_i = X, \;\; \emptyset \notin f_P(X) \tag{1}$$

An optimal parallelizing function should generate a large number of subproblems, that have few overlaps, and all objects of the same class should share at least one subproblem.

For splitting a problem into a set of subproblems, clustering with a cheap distance metric can be applied. We suggest to use soft-clustering instead of hard-clustering. This way an object can be placed in several subproblems. The main reason for using soft clustering is that parallelizing has to be fast such that

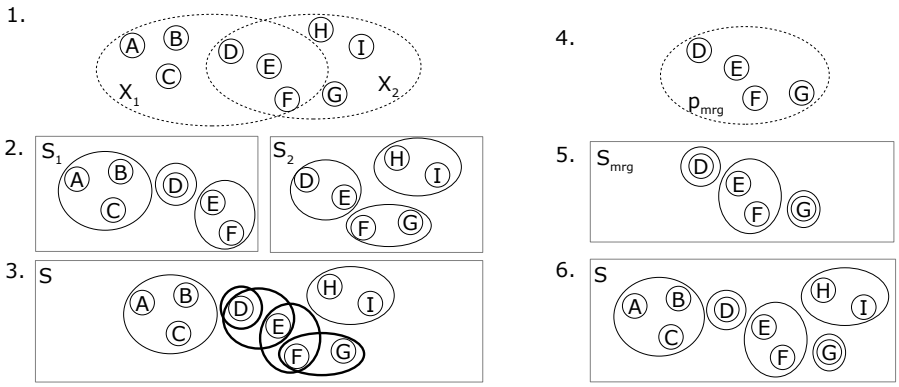

Fig. 4. Merge partial solutions: (1) A problem X was parallelized into overlapping sets X_1 and X_2. (2) Solutions S_1 for X_1 and S_2 for X_2 are predicted in parallel. (3) The combined solution S overlaps for objects D, E, F, G. (4) A new problem p_{mrg} is created for the overlaps and (5) a solution S_{mrg} is predicted. (6) Now the overall solution S is hard clustered.

decisions have to be as simple as possible. Especially at the borders of clusters, the degree of uncertainty is high. These serious decisions should not be made by a fast and approximative algorithm. With soft-clustering the algorithm can defer this decision to the more powerful central model.

A possible algorithm for parallelizing is clustering by canopies (see Algorithm 1). The design of this algorithm is inspired by the canopy blocker of McCallum et al. [10]. In contrast to the canopy blocker of McCallum et al., our Canopy-Clustering algorithm returns overlapping sets of objects. This way, the space complexity is $O(n)$ instead of $O(n^2)$. For Canopy-Clustering a cheap distance-function like TFIDF-cosine-similarity can be used. An efficient implementation should use an inverted index so that $Canopy(x)$ can be calculated quickly. When training data is available, optimal values for θ_{loose} and θ_{tight} can be found by maximizing both recall and reduction rate.

Merge Partial Solutions Using soft-clustering for parallelizing comes to the price, that solutions of subproblems may overlap and have to be merged. An example can be found in figure 4. Here, parallelizing puts the object D, E, F both in problem X_1 and X_2. The two central models predict different equivalences, i.e. the predicted clusters overlap. To solve these overlaps, we suggest to identify overlapping parts and collectively reestimate the class memberships of many overlapping objects. In step 3 of figure 4 the solution S is unsure about the equivalences of D, E, F and G. Thus, a new problem $p_{mrg} = \{D, E, F, G\}$ is created (figure 4, step 4) and is solved by an expensive central model (figure 4, step 5).

Our method for merging subsolutions iteratively eliminates overlaps until a hard clustered solution – that is a partition – is found. Inside each iteration, first

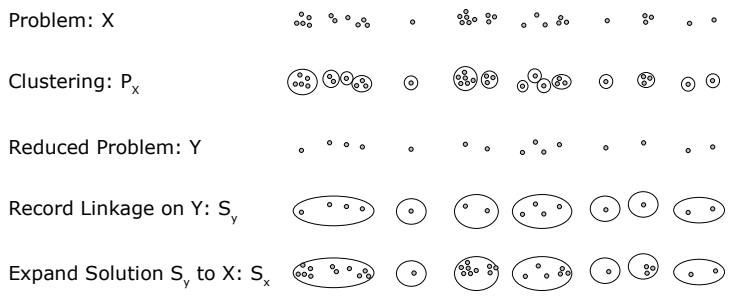

Fig. 5. Object reduction: Problem X is reduced to Y by clustering P_X and creating representatives for each cluster. Record linkage is performed on Y to create a solution S_Y. At last S_Y is expanded to S_X.

of all overlapping regions are identified and subproblems are generated. A new problem p is generated by picking a random object cluster c with overlaps and by extending it with other overlapping object clusters c'. Enlarging the problem p is stopped as soon as no other overlaps with this problem are found or the size of the problem extends a threshold θ_{mrg}. This threshold prevents the new problem to become too large and ensures that it can be solved by the central record linkage model. The set of subproblems P is extended until no more subproblems can be found. Then the subproblems are solved separately and afterwards are merged with the current solution. For finding maximal overlapping clusters, the overlap coefficient can be used:

$$overlap(c_1, c_2) = \frac{|c_1 \cap c_2|}{\min\{|c_1|, |c_2|\}} \tag{2}$$

It is easy to show that the proposed algorithm terminates and outputs a hard clustered solution. In each iteration at least one subproblem is generated out of two overlapping object clusters. After solving the subproblem with the central record linkage model, the solution to this subproblem contains no overlaps. So after each iteration the number of overlapping objects decreases.

4.3 Meta Model for Object Reduction

The second scaling technique targets large classes. If we look at problems with lots of classes, parallelizing is an efficient way to generate many subproblems, containing only a few different classes. But in problems with large classes, like in problems with Zipf distributed class sizes, another reduction step is necessary. The reason is that all objects of the largest class will be completely inside one subproblem – under the assumption that parallelizing was effective. In a problem with Zipf-distributed class sizes, the size k_{max} of the largest class is in $\Omega(n/\ln n)$, so this subproblem will have $\Omega(n^2/\ln^2 n)$ true pairs. Thus we suggest to reduce

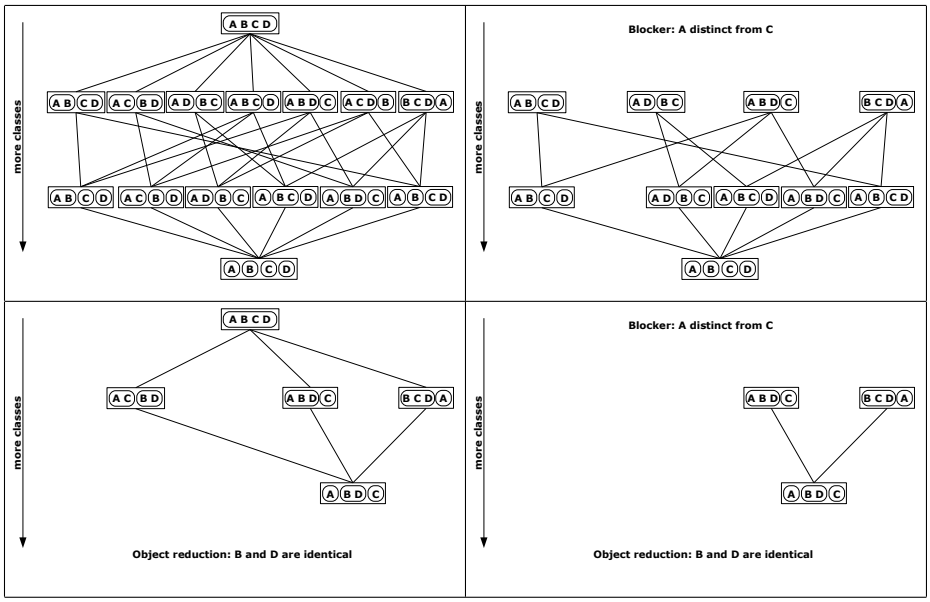

Fig. 6. Hypothesis space for a problem with four objects: A, B, C and D. A blocker reduces the space top-down – here the combination (A, C) is eliminated. Object reduction reduces the space bottom-up – here B and D are identified as identical. Combining object reduction and blocking results in a much smaller hypothesis space – here only three possibilities.

the problem size by eliminating pairs that are obviously identical. This will be done by merging these identical objects before applying an expensive model.

Method The overall method for solving a problem using object reduction is shown in figure 5. First we start with objects X. These objects are reduced to a subset Y. For this task one can use standard clustering techniques. In our experiments, we choose a HAC algorithm with complete-linkage and a very high threshold. As distance measure, we use the overlap coefficient over 2-grams. The objective of the reduction process is to produce a partition with perfect precision.

After having clustered X to a partition P_X, each cluster of P_X is regarded as an "object". The reduced object set Y composes of these objects. When sets of objects (= clusters of P_X) should be used as objects in a record linkage problem Y, the question arises how to represent each cluster by a single object. We propose to randomly pick one of the objects of each cluster in P_X, use it as a representative and put it into Y. Normally a random object of a cluster might not be a good representative because clusters might be diverse, but in our case clusters only contain very similar objects. Another approach would be to build prototypes, e.g. cluster centers and use them to built up Y.

Afterwards, the reduced problem Y is solved by an arbitrary record linkage model f_{OI}, that returns a solution on S_Y. The reduced solution S_Y is then expanded to all objects in X, so that a solution S_X results.

Object Reduction and Blockers When combining our object reduction method with a blocker, the set of potential hypotheses is reduced in two directions (see figure 6). The unrestricted hypothesis space contains all possible solutions, that is all partitions of X. A blocker reduces this space top-down by eliminating object combinations that are obviously different. Object reduction works bottom-up and searches for pairs that are very likely identical. Combining both reduction methods of blocking and object reduction results in a hypothesis space that only contains non-trivial hypotheses. Only these hypotheses have to be regarded by an expensive decision model.

Object reduction is effective particularly with regard to large classes, that appear in problems with non-uniformly distributed class sizes. In this case, it is very likely that many objects of a large class are very similar. This will result in many reductions and a smaller hypothesis space. With this reduction an expensive model can be applied to such problems.

It is important to note, that our proposed model for object reduction only performs the bottom-up step of figure 6. The blocking of obviously false pairs should be done by the central model of your choice. The reason is that blocking is already a standard technique in most models.

5 Evaluation

5.1 Dataset and Model Setup

In our experiments, we evaluate methods for scaling a state-of-the-art record linkage model. We evaluate on the *Cora* and the *Camera* dataset which are described in section 3. As expensive central model, we use the popular approach of training a probabilistic classifier and use it as a learned similarity measure for clustering the objects into sets of equal objects [6, 3, 16, 5]. Analogous to [5], the model uses constrained hierarchical agglomerative clustering with average linkage for collective decisions and as classifier a SVM (for *Cora*) and logistic regression (for *Camera*), respectively. As pairwise features over the textual attributes, this model uses several heuristic similarity measures, that are TFIDF-cosine-similarity, Overlap-coefficient over tokens, 2-grams and 3-grams; the model for the *Camera* dataset additionally uses some domain specific measures.

In each experiment, we randomly label 50% of the objects with their true class label and predict the whole dataset. We report the runtime and the F-Measure on the pairs between unlabeled objects. All experiments were run on a single standard PC. The parallel scaling method would considerably benefit from using more machines because each subproblem could be solved in parallel on another machine. Even though, also with the parallel scaling technique we only use one PC and solve all subproblem sequentially one after another on the same machine.

Table 2. Runtime and quality results for several scaling methods on the Cora dataset.

	Cora	
Scaling Method	F-Measure	Runtime (min)
None	0.948 ± 0.008	206
Blocking	0.954 ± 0.011	121
Object Reduction + Blocking	0.948 ± 0.011	52
Parallelizing + Blocking	0.936 ± 0.011	20
Parallelizing + Object Reduction + Blocking	0.944 ± 0.009	8

5.2 Comparison of Scaling Techniques

In the first experiment, we compare our two novel scaling techniques to the popular Canopy-Blocker [10]. As both parallelizing and object reduction are meta models, they can be used in compound models. In all we have five different scaling setups: (1) no scaling, (2) scaling by blocking, (3) scaling by parallelizing with blocking, (4) scaling by object reduction with blocking and (5) scaling by parallelizing and object reduction with blocking (see figure 2). We run all experiments five times with random train/ test splits.

Table 2 shows the average F-Measure quality with standard deviation and the average runtime for the five scaling approaches on the *Cora* dataset. As one can see, the runtime decreases from 121 minutes to 8 minutes when adding parallelizing and object reduction to a blocker based model. This corresponds to a speedup of 15 or in other words the runtime decreases by 93%. It is interesting, that parallelizing is so effective even though all subproblems are solved sequentially on the same machine. The reason is, that the cost of solving k small problems of the size n/k is much less than solving one problem of the size n.

On the other hand, our proposed scaling methods are also effective in terms of quality. Scaling the blocker based model with both parallelization and object reduction decreases the F-Measure only little from 95.4% to 94.4%. This difference is not statistically significant.

5.3 Scaling a Large Dataset with Parallelizing and Object Reduction

In the second experiment, we examine the components of the scaling framework in more detail on the *Camera* dataset with 15,481 objects and 956,957 true pairs.

Canopy-Clustering returned 90 subproblems and achieves a recall of 98%. Because a lot of these subproblems are very small, we automatically merged the smallest subproblems, so that each subproblem contains at least 200 objects. This is done, because we have to assure that in each subproblem is enough labeled data for training a pairwise model. Subproblems with more than 200 objects were not modified. In total 50 subproblems remain. In each subproblem object reduction is applied. Reduction achieved a precision of at least 98% in each subproblem. Afterwards the central model is applied on each reduced problem.

As Canopy-Clustering produces a lot of overlaps, merging the subproblems is no trivial task. The 50 subproblems contain in total 37,780 objects, that means on average each of the 15,481 objects is mentioned in 2.4 subproblems. The local models predict 3098 distinct classes in total for all subproblems. Still there are lots of overlaps, that have to be resolved by the merging process described in section 4.2. Merging needs 3 iterations to resolve all these overlaps and outputs a consistent solution. The F-Measure for the overall solution after all merging iterations is 93%.

The overall execution time was 8 hours and 10 minutes on a single machine. The runtime for splitting the problem into subproblems using Canopy-Clustering was about 5 minutes. Solving all 50 subproblems took 3 hours and 10 minutes. The largest subproblem was solved in less then 30 minutes. We run all parallelized subproblems one after another on a single machine, so runtime would decrease a lot if multiple machines were used in parallel. Theoretically the runtime for solving the subproblems in parallel can be lowered to 30 minutes using 7 machines of this type.

6 Conclusion and Future Work

We have shown that some important record linkage problems have Zipf-like distributed class sizes and that the standard technique of blocking does not scale with such problems. Thus we have proposed two techniques for scaling arbitrary record linkage models to large problems with non-uniformly distributed class sizes. The first one parallelizes a problem in many overlapping subproblems, so that each subproblem can be solved independently with an arbitrary record linkage model. Afterwards the solutions are merged by iteratively reestimating regions with uncertainty. The second scaling technique reduces the number of objects when dealing with large class sizes. By combining both techniques, record linkage models scale to problems having many classes as well as large classes. We have shown by experiments that our scaling techniques can scale a state-of-the-art record linkage model to challenging datasets and is efficient both in runtime and quality. As far as we know, our framework is the first approach that is able to efficiently solve large record linkage problems with both many classes and large class sizes.

One promising point for future work would be to develop other domain-independent parallelization methods that generate less overlaps than Canopy-Clustering. This might be achieved by transferring work on adaptive blocking [11, 12] to parallelizing.

Acknowledgements

This work was funded by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978.

References

1. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association **64** (1969) 1183–1210
2. Jin, L., Li, C., Mehrotra, S.: Efficient record linkage in large data sets. Proceedings of the 8th International Conference on Database Systems for Advanced Applications (DASFAA) (2003)
3. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003), Washington, DC (2003)
4. Culotta, A., McCallum, A.: Joint deduplication of multiple record types in relational data. In: CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management, New York, NY, USA, ACM Press (2005) 257–258
5. Rendle, S., Schmidt-Thieme, L.: Object identification with constraints. In: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM-2006), Hong Kong (2006)
6. Cohen, W.W., Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002), Edmonton, Alberta (2002) 475–480
7. Singla, P., Domingos, P.: Entity resolution with markov logic. In: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM-2006), Hong Kong (2006)
8. Newcombe, H., Kennedy, J., Axford, S., James, A.: Automatic linkage of vital records. Science **130** (1959) 954–959
9. Hernández, M.A., Stolfo, S.J.: The merge/purge problem for large databases. In: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data (SIGMOD-95), San Jose, CA (1995) 127–138
10. McCallum, A.K., Nigam, K., Ungar, L.: Efficient clustering of high-dimensional data sets with application to reference matching. In: Proceedings of the 6th International Conference On Knowledge Discovery and Data Mining (KDD-2000), Boston, MA (2000) 169–178
11. Bilenko, M., Kamath, B., Mooney, R.J.: Adaptive blocking: Learning to scale up record linkage. In: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM-2006), Hong Kong (2006)
12. Michelson, M., Knoblock, C.A.: Learning blocking schemes for record linkage. In: Proceedings of AAAI-2006. (2006)
13. Baxter, R., Christen, P., Churches, T.: A comparison of fast blocking methods for record linkage. In: Proceedings of the 2003 ACM SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation, Washington, DC (2003) 25–27
14. Christen, P., Churches, T., Hegland, M.: A parallel open source data linkage system. In: Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-2004), Sydney (2004)
15. Karypis, G., Kumar, V.: Parallel multilevel graph partitioning. In: Proceedings of the 10th International Parallel Processing Symposium (IPPS-1996). (1996)
16. Bilenko, M., Basu, S., Sahami, M.: Adaptive product normalization: Using online learning for record linkage in comparison shopping. In: Proceedings of the 5th IEEE International Conference on Data Mining (ICDM-2005). (2005)