Universiteit Utrecht

Computing Science

Algorithmic data analysis

Experimental comparison of heuristic cluster-editing algorithms for entity deduplication.

Author:
Mohamed Ali Addi

Supervisors:
dr. Ad Feelders
prof. dr. Mehdi Dastani

Thesis-number:
ICA-5669405

April, 2022

Abstract

The application of heuristic weighted cluster-editing algorithms within the scope of entity deduplication is a relatively unexplored area. This research has aimed at comparing the efficacy of different heuristics on both real-world and artificially-generated entity-deduplication data-sets. The research has shown that the *Force*, *Spectral*, *Vote/BOEM*, and *Split-Merge* heuristics perform relatively well for precision in comparison to the benchmark heuristics *Pivot* and *Closure* on a variety of data-sets. The research has also shown that there is a correlation between the objective-function of these heuristics and the F_1-score, hence giving merit to the idea of using the objective-function as a proxy for the F_1-score in the absence of a ground-truth.

Contents

Chapter 1

Introduction

Data is being collected in larger volumes, increasing speeds and more variety [1]. All of that collected data has been of great use to society. It has given us much insight into how the world around us works, and along with it an incentive to collect even more data, even faster. This has resulted in an immense landscape of datasets. Datasets which are often generated by separate parties, even if the data represents the same real-world domain.

Analysis of separate datasets that refer to the same real-world domain is less interesting than the analysis of combined datasets. This is because insights which would only be apparent through analysis of merged datasets are then overlooked. This has created a need to merge datasets concerning the same real-world domains. Separate datasets can contain data that refers to the same real-world objects, which makes it necessary to remove/merge any duplicates in the merged dataset. This problem is also known as entity deduplication [2].

In practice duplicate entities do not have to be completely identical. This makes the problem of entity deduplication more difficult than finding entities which are exactly identical. A naive deduplication approach would be to evaluate all pairs of entities with a similarity measure. Then the duplicate pairs are those which score above a predefined similarity threshold. For a dataset of n entities, this approach would require $O(n^2)$ comparisons. The magnitude of modern day datasets makes this exhaustive approach too costly to perform within practical running times.

Entity deduplication becomes a more practical task if the comparison of all pairs can be avoided, for example through a blocking mechanism. A blocking mechanism divides the dataset into 'blocks'. Pairwise comparisons are only performed for entities within the same block [3].

Even with a good blocking mechanism in place, there still remains another issue which is explained by the following example. Consider a dataset consisting of entities $\{A, B, C, D\}$. The pairwise similarities for the entities are given in table 1.1. Suppose that the similarity threshold is set at zero. Deduplication of the dataset with the above mentioned approach would then result in the pairs (A, B) and (A, C) to be deemed duplicates. The issue here however is that this deduplication violates transitivity. If (A, B) are the same, and (A, C) are the same, then it should follow that (B, C) are the same.

	A	**B**	**C**	**D**
A	–	0.3	0.8	-0.8
B	0.3	–	-0.1	-0.9
C	0.8	-0.1	–	-0.7
D	-0.8	-0.9	-0.7	–

Table 1.1: Pairwise similarities for entities *A,B,C and D*

These duplicate relationships within a dataset can be represented as a graph. In figure 1.1 the entities are represented as nodes, and a link between nodes represents that the two entities are duplicates. In order to make the the deduplication conform with the transitivity constraint, a link should be present for the pair (B, C). Put differently, the graph should be transformed into a collection of disjoint cliques. In the unweighted case this transformation should be done with as few insertions and deletions of links as possible.The problem of finding such a transformation is called the *cluster-editing* problem [4].

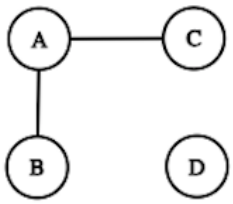


Figure 1.1: Graph representation where an edge implies the 'same as' relation.

There are exact algorithms for solving the weighted cluster editing problem through different approaches [5]. These exact algorithms do generate optimal solutions, but they do have impractical running times. These exact algorithms tend to edit clusters within practical time only if a dataset is relatively small. This is a major bottleneck since real-world datasets tend to be quite large.

Given the inefficiency of these exact algorithms, there is a need to look further for more efficient means to cluster-edit such graphs. There are heuristic algorithms which tend to perform significantly more efficient. The application of such heuristics as a post-processing step for entity deduplication remains relatively unexplored however. Therefore it is the primary focus of this research.

Chapter 2

Research question

2.1 The weighted cluster-editing problem

Weighted Cluster-editing is the problem of transforming a graph into a collection of disjoint cliques while minimizing the costs of edge deletions/insertions. The costs are the sum of the weights of the edges which are removed, minus the sum of the weight of the edges which are inserted [5]. The problem is defined formally in definition 1 .

Definition 1 *Given the weighted undirected graph $G = (V, E, w)$ with the edge set $E := \{(i, j) : w(i, j) > 0\}$ and the similarity function $w : V^2 \to \mathbb{R}$, compute $\delta(G) := min(cost(G \to G^*))$ where G^* is a transitive undirected graph with $costs(G \to G^*) = \delta(G)$ where the costs are defined as:*

$$costs(G \to G^*) := \sum_{(i,j)\in(E\setminus E^*)} w(i, j) - \sum_{(i,j)\in(E^*\setminus E)} w(i, j).$$

The problem is perhaps better understood through an example of an unweighted instance. The unweighted case can also be seen as a weighted case in which weights are either 1 or -1. Here $w(i, j) = 1$ should be interpreted as "i and j are similar" and $w(i, j) = -1$ as "i and j are dissimilar". Given the unweighted undirected graph $G = (V, E)$ in figure 1.1, cluster-editing transforms G into a set of disjoint cliques while minimizing costs. Since the graph is unweighted, all edge modifications cost the same. Figure 2.1 shows three optimal solutions for the unweighted graph in 1.1. Note that the optimal solution is not unique in this case since any of the three modifications cost the same.

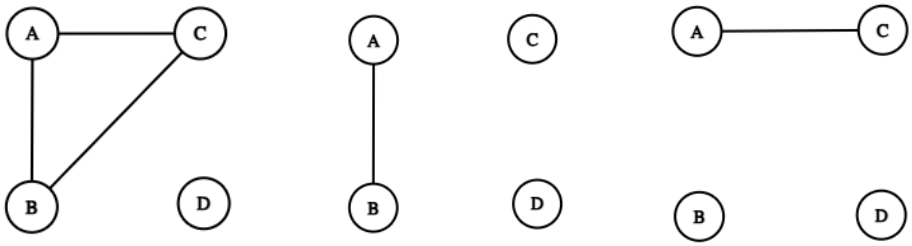


Figure 2.1: Three optimal cluster-edit solutions for the unweighted case for the graph in figure 1.1

Given the weights w present in table 1.1 the goal of weighted cluster-editing remains the same as the unweighted case. Though the costs for modifying edge (i, j) now depends on the weight/similarity of the edge. The insertion of the edge (B, C) costs 0.1, the deletion of the edges (A, C) and (A, B) cost 0.8 and 0.3 respectively. Therefore making the insertion of edge (B, C) the optimal solution for the weighted case.

2.2 Research question

The goal of this research is to determine whether heuristic weighted cluster-edit algorithms are an effective post-processing step in entity deduplication. This goal consists out of different aspects and therefore gives us multiple research sub-questions.

We are ultimately interested in how well the clustering produced by a heuristics corresponds to the true clusters. This is typically measured by the F_1-score. The ground truth is needed for the F_1-score, which may not always be available. However, if there is a correlation between costs and the F_1-score, then costs can be used as a proxy for the F_1-score. This leads us to the following two questions:

Question 1

Question 1.1
Which heuristic algorithms for weighted cluster-editing perform best?

Question 1.2
What is the relation between the value of the objective function and the F_1-score?

A significant part of the research focuses on the problem of entity deduplication. It could be the case that entity deduplication problems have different properties than clustering problems in general. The distribution of cluster sizes, and the distribution of intra-cluster weights and inter-cluster weights could vary. This means that it is still unsure whether best performing algorithms for clustering problems in general will perform best for entity deduplication as well. Knowledge of these properties will allow us to generate artificial data for entity deduplication problems in order to determine the performance of different algorithms in a systematic way. This leads to the two questions:

Question 2

Question 2.1
Which heuristic algorithms for weighted cluster-editing perform best for the entity-deduplication problem?

Question 2.2
What are the typical properties of entity deduplication problems in terms of the distribution of cluster sizes and weights?

2.3 Area of impact

Entity deduplication can be useful in domains in which different knowledge graphs are frequently used to answer questions that depend on the analysis of such graphs. A prime example of this is the domain of historians which often times need to deal with knowledge graphs governed by different institutions. In this case different URI's can represent the same real world objects. Though since these URI's are not unified, critical insights can be overlooked by inference software simply because the entities across different knowledge graphs have not been matched [6]. The research can potentially help with uncluttering the ever growing landscape of big data.

Chapter 3

Cluster-edit Algorithms

A fair amount of research has been done in the field of weighted cluster-editing. Research has shown that heuristic weighted cluster-editing algorithms tend to perform relatively well in comparison to exact algorithms, though in much less time [5]. In this chapter an overview of state-of-the-art heuristic weighted cluster-editing algorithms is presented, alongside an exact algorithm which will be used for comparative purposes on smaller datasets.

The heuristics included in this chapter are *Force* [5], *Split-Merge*[7], *Spectral*[8], *Vote/BOEM*[9], *Pivot* [9], and *Closure*[10]. The exact algorithm is an *Integer Linear Programming* formulation of the problem [6].

3.1 Exact weighted cluster-edit algorithm

An Integer Linear Programming formulation (ILP) is used to find an optimal cluster-editing solution. The ILP approach is an exact cluster-edit algorithm which formulates the problem as the objective function and the set of linear constraints in definition 2.

Definition 2

$$\min_{\boldsymbol{X}} \sum_{(i,j)\in\Lambda} w(i,j) \; - \sum_{1\leq i<j\leq n} w(i,j)x_{ij} \tag{1}$$

$$\begin{aligned}
\text{subject to} \quad & +x_{ij} + x_{jk} - x_{ik} \leq 1 \text{ for all } 1 \leq i < j < k \leq n, && (2)\\
& +x_{ij} - x_{jk} + x_{ik} \leq 1 \text{ for all } 1 \leq i < j < k \leq n, && (3)\\
& -x_{ij} + x_{jk} + x_{ik} \leq 1 \text{ for all } 1 \leq i < j < k \leq n, && (4)\\
& x_{ij} \in \{0,1\} \qquad \text{for all } 1 \leq i < j \leq n && (5)
\end{aligned}$$

The constraints model which edges between pairs of entities are present in the solution. The constraints (2)-(4) enforce that the solution is transitive. The objective function (which is to be minimized) is the sum of positive weights minus the sum of the weights of the edges which are present in the solution. Λ is the set of pairs which have a positive similarity $w(i,j)$. The decision variables x_{ij} take on the value of one if the edge between (i,j) is present in the solution, and zero otherwise.

This approach starts to take too much time to be of practical use as the number of entities surpasses a few hundred. Real-world entity deduplication datasets tend to be much larger and contain relatively few duplicates however [11]. Making the ILP approach not practically suitable due to its inefficiency.

3.2 Force

Force is a layout based heuristic that is inspired by physical intuition in which entities are seen as magnets, and edges as rubber bands. Entities are placed arbitrarily on a the circumference of a circle in $\mathbb{R}^2$. Due to the attraction of the rubber bands and the magnetic repulsion, the system moves to a state of minimal energy. This places entities which are relatively similar close to one another, and those which are not relatively far apart from one another [5]. The edge modifications are then made based on the positions of the entities in the minimal energy state.

The heuristic consists out of the following three phases: a layout phase, a partitioning phase, and post-processing phase. These three are briefly described below, followed by a more in depth description of each phase along side the algorithms pseudo code which can be found in Algorithm 1.

- *Layout phase:* Over R iterations the attractive and repulsive forces are applied to each entity, placing similar entities close to one another in $\mathbb{R}^2$.
- *Partitioning phase:* Based on the layout of the initial phase, entities are clustered[1] together through single-linkage clustering based on euclidean distance.
- *Post-processing phase:* Clusters are repeatedly merged if it reduces costs. Entities are repeatedly moved to a different cluster if it reduces costs.

Layout phase

Given a set of n entities O, each entity o_i is initially placed at a random position $pos[i]$ on a circle with radius p_0 around the origin in $\mathbb{R}^2$. The distance $d(i, j)$ between two entities o_i and o_j is defined as the distance between $pos[i]$ and $pos[j]$.

This is followed by R iterations in which for each entity o_i, $pos[i]$ is updated based on the position of all other entities, their respective similarities and the number of iterations r that have already passed. For each pair (o_i, o_j) where $i \neq j$ $pos[i]$ is moved towards $pos[j]$ if their similarity $w(i, j)$ is positive, and away from $pos[j]$ if otherwise. The amount with which $pos[i]$ is updated is scaled by the similarity and distance between the two entities, and the number of iterations that have passed. This results in the positions being updated by smaller steps as the number iterations increases.

[1] A cluster of entities translates to the presence of an edge in the graph between all pairs of entities within the same cluster.

The unit vector along which the positions are moved is defined as follows:

$$\frac{pos[j] - pos[i]}{d(i,j)} \cdot Min(M(r), amount)$$

At each round r the magnitude of the unit vector is capped by the function $M(r)$ which takes on the following value:

$$n \cdot M_0 \cdot (1/(r+1))^2$$

Here M_0 is a user-defined scaling factor. If $w(i,j) > 0$ the unit vector is scaled by the following amount:

$$(f_{att} \cdot Fatt(d(i,j)) \cdot w(i,j))$$

Here $F_{att}(d(i,j))$ is equivalent to $log(d(i,j)+1)$, and f_{att} is a user-defined scaling factor. If $w(i,j) < 0$ the unit vector is scaled by the following amount:

$$(f_{rep} \cdot F_{rep}(d(i,j)) \cdot |w(i,j)|)$$

Here F_{rep} is defined as the inverse of F_{att} and f_{rep} is a user defined scaling factor. Here too the magnitude of the vector is capped by the value of $M(r)$ for the r^{th} round.

Partition phase

During this phase, the layout from phase one is used to cluster the entities through single linkage clustering. Entities o_i and o_j are assigned to the same cluster given a distance parameter δ if there exists a path $i = i_0, i_1, ..., i_k = j$ such that the distance between any two consecutive nodes in the path is at most δ. This process is repeated for all deltas within the range $[\delta_{init}, \delta_{max}]$ with step size σ. The clustering resulting in the best costs throughout this process is the resulting clustering of this phase.

Post-process phase

Given the clustering resulting from phase two, each pair of clusters is evaluated. If merging two clusters results in lower costs, then they are merged. This is repeated until no such moves exist any longer. This is followed by evaluating for each entity if moving it to another cluster results in lower costs. If so, the entity is moved to the respective cost lowering cluster. This is repeated until no such cost-lowering moves exist.

Algorithm 1 Force-heuristic

```
1:  procedure LAYOUT
2:      for Node i in graph do
3:          pos[i] ← random point at distance p_0 from origin
4:      for r = 1, ..., R do
5:          F_att := log(d + 1)
6:          F_rep := 1/F_att
7:          for Node j in graph where i ≠ j do
8:              M(r) ← n · M_0(1/(r + 1))^2
9:              if w(i, j) > 0 then
10:                 amount ← (f_att · F_att(d(i, j)) · w(i, j))
11:                 pos[i] ← ((pos[j] − pos[i])/d(i, j)) · Min(M(r), amount)
12:             if w(i, j) < 0 then
13:                 amount ← (f_rep · F_rep(d(i, j)) · |w(i, j)|)
14:                 pos[i] ← ((pos[j] − pos[i])/d(i, j)) · Min(M(r), amount)
15:     return pos

16: procedure PARTITION
17:     δ ← δ_init
18:     BestClustering ← EmptyClustering
19:     f_σ ← 1.1
20:     σ ← 0.01
21:     while δ < δ_max do
22:         CurrentClustering ← SingleLinkageCluster(nodes, δ)
23:         if Costs(CurrentClustering) < Costs(BestClustering) then
24:             BestClustering ← CurrentClustering
25:         δ ← δ + σ
26:         σ ← σ · f_σ
27:     return BestClustering

28: procedure POSTPROCESS
29:     BestClust ← CurrentClust
30:     while a cost lowering merge exists do
31:         for each pair of clusters, C_i, C_j do
32:             MergeIfLowersCosts(C_i, C_j)
33:             CurrentClust ← MergeClusters(C_i, C_j)
34:             if Costs(CurrentClust) < Costs(BestClust) then
35:                 BestClust ← CurrentClust
36:     while a cost lowering move exists do
37:         for each node i do
38:             for each cluster C, with i ∉ C do
39:                 CurrentClust ← MoveToCluster(i, C)
40:                 if Costs(CurrentClust) < Costs(BestClust) then
41:                     BestClust ← CurrentClust
42:     return BestClust
```

3.3 Vote/BOEM

The Vote/BOEM heuristic consists out of two phases. The vote phase creates an initial clustering, and the BOEM phase serves as a score improving post-processing step.

The VOTE algorithm adds each entity to the cluster which maximizes the total net weight. If no cluster exists for which the total net weight gain is positive, the entity becomes a new cluster. The resulting clustering is dependent on the order in which the entities are processed. The pseudo-code of the algorithm can be found in algorithm 2.

Algorithm 2 VOTE

```
1:  procedure VOTE
2:      k ← 0                                          ▷ Number of created clusters
3:      for i = 1, n do
4:          for c = 1, k do
5:              Quality_c ← Σ_{j∈C[c]} w(i, j)
6:          c* ← argmax_{1≤c≤k} Quality_c
7:          if Quality_{c*} > 0 then
8:              C[c*] ← C[c*] ∪ {i}
9:          else
10:             C[k++] ← i                             ▷ A new cluster is created
11:     return C
```

On the resulting clustering, the *BOEM (Best One Element Move)* algorithm is executed. It repeatedly makes the move which improves the score the most. The allowed moves are removing a vertex from a cluster and moving it to another cluster or turning it into a new cluster. Such moves are repeated until a local optimum is reached or a user-defined k_{max} iterations are performed[9].

3.4 SplitMerge

The SplitMerge heuristic generates a clustering in three phases: the generation of an initial clustering based on the connected components of the input graph, splitting the resulting clusters to ensure that entities within the same cluster are similar, and lastly merging clusters which are highly similar [7]. Each phase is outlined below.

Connected components phase

Given the undirected weighted input graph $G = (V, E, w)$, the entities within each connected component formed by the positively weighted edges become a cluster. All other entities become singleton clusters.

Split phase

An entity is said to be insufficiently similar if the average similarity to other cluster members is below the user-defined threshold t_s. Given the clustering from phase one, the entities within a cluster which are insufficiently similar become singleton clusters. This process is repeated until no such entities are present in the graph. Afterwards, each cluster has a cluster-representative assigned. The cluster-representative is the 'medoid' of such a cluster (i.e. the entity with the highest average similarity to all other nodes in the cluster).

Merge phase

Two clusters are highly similar if the similarity between their respective representatives is above the user-defined threshold t_m. Given the output of the split phase, the most highly similar pair of clusters is merged. The new cluster then has its representative updated. This process is repeated until there are no highly similar pairs left.

3.5 Spectral partitioning

Given a weighted undirected graph $G = (V, E, w)$, spectral partitioning determines which edges to delete in order to partition V into two disjoint sets U and W. These sets (i.e. clusters) have high intra-cluster similarities, and low inter-cluster similarities between the entities [8].

The set of to be deleted edges is also known as the ratio-cut. The ratio-cut is obtained through manipulation of the Laplacian matrix L of G, which defined as $D - A$. Where D is the degree matrix of G, and A is the weighted degree matrix G (where A_{ij} corresponds with $w(i, j)$). The minimal ratio-cut is obtained through the second smallest eigenvalue (named the Fiedler value) of L. The sign of each value in the Fiedler vector determines whether an entity belongs to set U or W [12]. The ratio-cut minimizes the following ratio:

$$\frac{|E(G) \cap (U \times W)|}{|U| \cdot |W|}$$

Spectral partitioning determines how to split a graph into two clusters but does not determine how many clusters the graph should be partitioned into. The number of clusters k is determined through the *eigengap* heuristic. The heuristic determines k based on the largest difference in consecutive eigenvalues of L [13]. For example, suppose that the largest eigengap lies between the 4^{th} and 5^{th} eigenvalue, then k would be set at four. The standard spectral partition method is then applied to the k eigenvectors corresponding to the first k eigenvalues. The end result is a clustering with k clusters.

3.6 Pivot

The Pivot heuristic serves as a baseline to compare the other heuristics with. Suppose we have a weighted undirected graph $G = (V, E, w)$, the heuristics repeatedly takes an unclustered vertex at random and creates a new cluster containing that vertex and all its unclustered neighboring vertices with a positive weight [14].

3.7 Closure

This heuristic is also a baseline for comparison. The heuristics is rather straightforward. Suppose we have a weighted undirected graph $G = (V, E, w)$, and the connected-components which result from either the *knn* blocking mechanism or simply those which are present in the graph consisting of edges with a positive weight. The transitive-closure heuristic simply treats each connected component as a cluster, by placing an edge between all the nodes within the same connected-component [10]

Chapter 4

Method

4.1 Experimental aims

Entity-deduplication in varying data One of the chief aims of this research is to asses how well heuristic weighted-cluster editing algorithms perform at the task of correlation-clustering. The datasets that are available can vary from one another with respect to their properties such as size, the number of duplicates, the degree of similarity between duplicate- and non-duplicate entities etc. Therefore the experiments will concern themselves with datasets (both real and artificial) with different properties in order to get a sense of the performance of the heuristics in different situations.

Characterising Entity-deduplication data To get a sense of what characterises an entity-deduplication dataset, various properties of each real-world dataset are computed. Properties such as dataset-size, cluster-size distributions, inter- and intra-cluster pairwise similarity distributions. Together, these properties characterises a dataset.

Generating artificial data The experiments also aim to investigate the heuristic performance on data which vary from the observed ranges in real-datasets. Therefore artificial data is generated to meet these needs. Given the scarcity of publicly available entity-deduplication data, artificial data which resembles the properties of real-data is generated as well.

Measuring performance through costs Each dataset used in the experiments has an available ground-truth in the form of labeled pairs of duplicates. The ground-truth can be used to asses the performance through evaluation metrics such as the *F1-score* . In practice such ground-truth is often (partially) unavailable, and in cases in which it is available it would make the task of cluster-editing obsolete. Only the objective-cost function is always available to asses performance. The experiments will therefore also look at the relationship between costs and performance metrics in order to asses its suitability as a proxy for ground-truth based metrics.

4.2 Data description

4.2.1 Data properties

A dataset can be represented by the following set of properties. Each dataset represents N entities of which a subset of pairs are duplicates. A similarity function $s : (e_i, e_j) \rightarrow \mathbb{R}$ determines the degree to which the pair of entities (e_i, e_j) is similar. The ground truth is represented as a set duplicate pairs. Entities that are said to be duplicates of one another can be seen as a cluster. The set of similarities between entities in the same cluster and those in different clusters can be represented as probability distributions. The same goes for the set of cluster sizes.

4.2.1.1 Cluster-size distribution

The distribution of cluster sizes is represented as a discrete probability distribution. Given the ground truth of duplicate pairs, the cluster sizes are determined as follows. The graph representation of the duplicate entities consists of a node for each entity. Each duplicate pair is represented as an edge between the two nodes. Each clique within the graph represents a cluster. The sizes of these cliques represent the discrete cluster size distribution for a dataset.

4.2.1.2 Similarity distributions

Given a pairwise similarity function, a dataset can be characterised by two continuous probability distributions representing the similarities for pairs of entities that are within the same cluster and those that are in different clusters; aptly named *intra-* and *inter-cluster* similarity distributions.

The similarity measure used to express how similar a pair of entities is should ideally indicate that a pair is relatively dissimilar if they are not in the same cluster, and relatively similar if they are in the same cluster. To meet these criteria two different similarity measures are used in the experiments depending on the dataset. Namely the *Cosine similarity* and *Approximate string matching*.

Cosine similarity Given a pair of entities (A, B) where each entity is represented by a vector, the Cosine Similarity assigns a value within the range $[-1, 1]$. A score of 1 indicates perfect similarity, whereas -1 indicates perfect dissimilarity. The value represents the angle between the two vectors. Cosine Similarity is defined as the dot product of the two vectors divided by the product of the magnitude of the two vectors.

$$Cosine(A, B) = \frac{A \cdot B}{\|A\| \times \|B\|}$$

Approximate string matching Cosine similarity is not suitable for the direct comparison of strings. Direct comparison of letters and their positioning is not suitable either, since the sentences *"Clustering is the grouping of objects together"* and *"Grouping objects together is called clustering"* would not be considered similar.

Approximate String matching uses the so called *Levenshtein Distance* to measure the differences between the sequences and patterns present in a pair of strings. Given the pair of strings (a, b) where $|a|$ and $|b|$ are their respective lengths, the tail of the string is all but the first character of the string, the metric is defined definition 3.

Definition 3

$$\text{lev}(a,b) = \begin{cases} |a| & \textit{if } |b| = 0 \\ |b| & \textit{if } |a| = 0 \\ \text{lev}(\text{tail(a)}, \text{tail(b)}) & \textit{if } a[0] = b[0] \\ 1 + \min \begin{cases} \text{lev}(\text{tail(a)}, \text{b}) \\ \text{lev}(\text{a}, \text{tail(b)}) \\ \text{lev}(\text{tail(a)}, \text{tail(b)}) \end{cases} & \textit{otherwise.} \end{cases}$$

The approximate matching is defined as the mean of the partial ratio, and ratio comparison of the two strings. This results in a value within the interval $[0, 1]$. The ratio comparison is defined as the Levenshtein distance between the two entire strings. The partial ratio is defined as the comparison of the shortest of the two strings with all the sub-string of the other string of the same size.

Estimating the similarity distributions The similarity distributions have been fit to various continuous probability distribution. None showed a statistically significant fit, though the most significant fit was the *β-distribution*. The distribution is bound to the interval $[0, 1]$, and can be scaled to match the interval of the cosine similarity. The Beta distribution has its shape defined by two parameters, α and β. The chief aim is to find parameter estimates for the two distributions which represent the intra- and inter-cluster similarity distributions. For this the set of similarities between duplicate pairs S_{intra} is used, and a set of similarities of randomly sampled pairs from the dataset S_{inter}.

Given both sets of similarities, the shape parameters are estimated using the method of maximum likelihood. The Kolmogorov-Smirnov test is applied to compare the sample distribution with the distribution derived from the maximum likelihood estimation. This statistical test results in a p-value for the null hypothesis that states that the samples are drawn from the reference distribution.

4.2.1.3 Distribution overlap

The datasets can be characterised by the overlap of the inter- and intra-cluster similarity distributions. The overlap of these distributions is determined through the properties of their respective Beta distributions. $f(x)$ represents the probability density function of the β-distribution. Given shape parameters α and β, it is defined as a power function of x. Here $B(\alpha, \beta)$ is defined as the product of the gamma functions of α and β divided by the gamma function of the sum of α and β. Defined more formally:

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \qquad B(\alpha,\beta) = \frac{(\alpha-1)!(\beta-1)!}{(\alpha+\beta-1)!}$$

Given an inter- and intra-cluster Beta distributions S_{inter} and S_{intra}, represented respectively by the pairs of shape parameters (α_1, β_1) and (α_2, β_2). Suppose that x is the intersection point of the probability density functions of S_{inter} and S_{intra}. The measure of overlap of the two distributions is defined by $P(X \leq x)$ for S_{intra} and $P(X \geq x)$ for S_{inter}. In more intuitive terms this represents to what degree a dataset contains pairs of entities for which it is not possible to determine whether they are duplicates or not based on the provided similarity measure.

4.2.2 Real-world data

A total of 5 real-world datasets are used in the experiments: *Golden agents, Affiliations, Geographical settlements, Music Brainz, and North-Carolina Voters.*

The *Golden agents* dataset stems from research conducted in [15]. The remaining datasets stem from [16], which is a collection of benchmark entity-clustering datasets made available by the university of Leipzig.

4.2.2.1 Golden agents

The *Golden agents* dataset represents records of people from the Dutch Golden ages. In the dataset a number URI pairs refer to the same real-world person. Though their respective node-embeddings may differ due to errors such as misspelled names and uncertain dates etc. The dataset consists of 4553 entities. Information about its cluster-size distribution, and intra- and inter-cluster similarity distributions can be found in the tables 4.1, 4.2, and 4.3.

dataset	***cluster sizes***	***probabilities***
Golden agents	1-4	(0.73, 0.249, 0.02, 0.001)

Table 4.1: Golden agents cluster size distribution

dataset	***alpha***	***beta***	***p_value***
Golden agents	89.21	0.96	3.69e-08

Table 4.2: intra-cluster similarity β distribution for golden agents

dataset	***alpha***	***beta***	***p_value***
Golden agents	12.99	6166299.49	1.48e-15

Table 4.3: inter-cluster similarity β distribution for golden agents

Figure 4.1 shows the histograms for the inter- and intra-cluster similarities for a set of samples from the approximated distributions, and the set of similarities that have been sampled from the data-set. It is apparent in tables 4.2, and 4.3 that the p-values for the approximated distributions are small enough to reject the null-hypothesis. Which means that we reject that the similarities stem from the approximated distribution. However, the histogram does show that the real and approximated distributions have similar shapes.

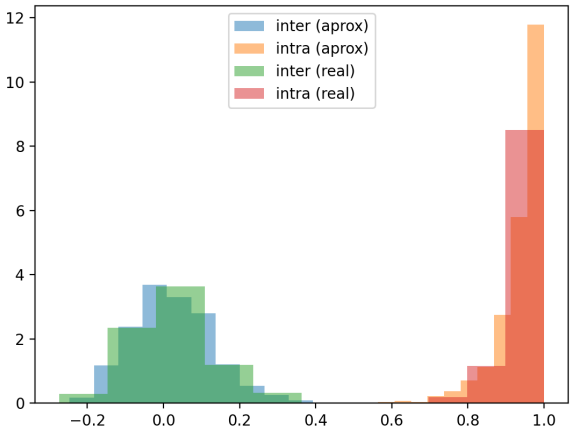


Figure 4.1: A histogram of the approximated β-distributions for the inter- and intra-cluster similarities for the *Golden-Agents* dataset. The β-distribution is scaled to the range between $[-1, 1]$, and the sampled distribution is also plotted for reference.

4.2.2.2 Affiliation strings

The *Affiliations* dataset contains string extracted from database publications in the period 2000-2009. These affiliations strings consist of multiple defining parts such as the institution name and city. The dataset consists out of 2260 entities. Information about its cluster-size distribution, and intra- and inter-cluster similarity distributions can be found in the tables 4.4, 4.5, and 4.6.

dataset	***cluster sizes***	***probabilities***
Affiliation strings	2-4	(0.51, 0.29, 0.2)

Table 4.4: Affiliation strings cluster size distribution

dataset	***alpha***	***beta***	***p_value***
Affiliation strings	4.21	2.41	0.00019

Table 4.5: intra-cluster similarity β distribution for affiliation strings

dataset	***alpha***	***beta***	***p_value***
Affiliation strings	5.26	20.39	4.70e-33

Table 4.6: inter-cluster similarity β distribution for affiliation strings

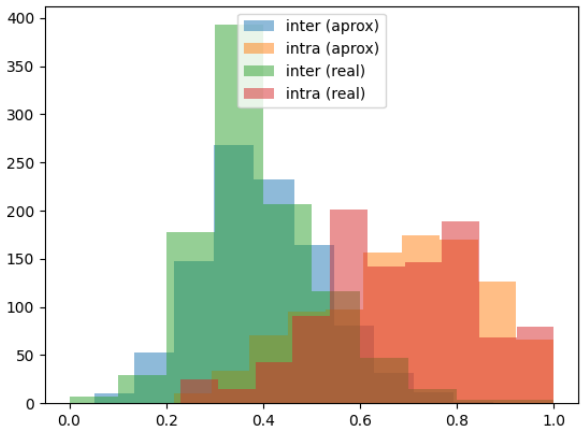


Figure 4.2: A histogram of the approximated β-distributions for the inter- and intra-cluster similarities for the *Affiliation strings* dataset. The β-distribution is scaled to the range between $[-1, 1]$, and the sampled distribution is also plotted for reference.

4.2.2.3 Geographical settlements

The *Geographical settlements* dataset contains information about real-world geographical entities from four different sources.The dataset consists of 3054 entities. Information about its cluster-size distribution, and intra- and inter-cluster similarity distributions can be found in the tables 4.7, 4.8, and 4.9.

dataset	***cluster sizes***	***probabilities***
Geographical settlements	1-4	(0.040, 0.053, 0.042, 0.865)

Table 4.7: Geographical settlements cluster size distribution

dataset	***alpha***	***beta***	***p_value***
Geographical settelments	3.20	0.84	0.0

Table 4.8: intra-cluster similarity β distribution for geographical settlements

dataset	***alpha***	***beta***	***p_value***
Geographical settelments	29.66	2659120.09	5.25e-12

Table 4.9: inter-cluster similarity β distribution for geographical settlements

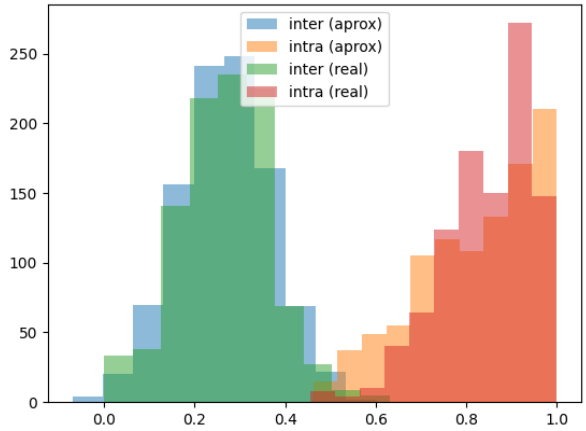


Figure 4.3: A histogram of the approximated β-distributions for the inter- and intra-cluster similarities for the *Geographical settlements* dataset. The β-distribution is scaled to the range between $[-1, 1]$, and the sampled distribution is also plotted for reference.

4.2.2.4 Music-Brainz

The *Music-Brainz* dataset is a semi-synthetic dataset based on real song-record data.The dataset consists of 5000 entities. Information about its cluster-size distribution, and intra- and inter-cluster similarity distributions can be found in the tables 4.10, 4.11, and 4.12.

dataset	***cluster sizes***	***probabilities***
Music-Brainz	1-4	(0.81, 0.166, 0.023, 0.001)

Table 4.10: Music-brainz cluster size distribution

dataset	***alpha***	***beta***	***p_value***
Music-brainz	75247751.40	4.29	0.0

Table 4.11: intra-cluster similarity β distribution for music-brainz

dataset	***alpha***	***beta***	***p_value***
Music-brainz	0.379	135.88	0.0

Table 4.12: inter-cluster similarity β distribution for music-brainz

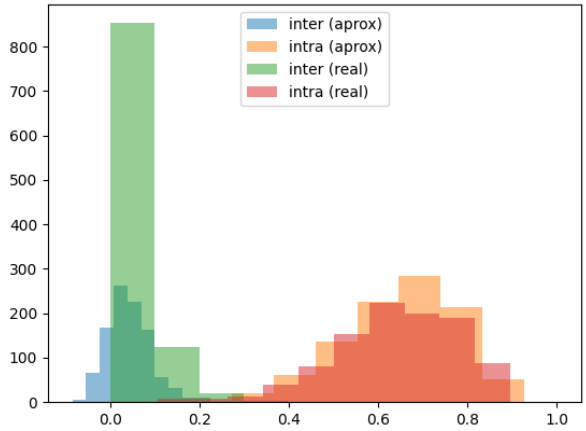


Figure 4.4: A histogram of the approximated β-distributions for the inter- and intra-cluster similarities for the *Music-brainz* dataset. The β-distribution is scaled to the range between $[-1, 1]$, and the sampled distribution is also plotted for reference.

4.2.2.5 North-Carolina Voters

The *North-Carolina voters* dataset consists of records from the North-Carolina voter registry and has been extended with artificial data with similar attribute values. The dataset consists of 10000 entities. Information about its cluster-size distribution, and intra- and inter-cluster similarity distributions can be found in the tables 4.13, 4.14, and 4.15.

dataset	***cluster sizes***	***probabilities***
North-Carolina voters	1-2	(0.9982, 0.0018)

Table 4.13: North-Carolina voters cluster size distribution

dataset	***alpha***	***beta***	***p_value***
North-Carolina_voters	1.29	0.89	5.49e-07

Table 4.14: intra-cluster similarity β distribution for North-Carolina

dataset	***alpha***	***beta***	***p_value***
North-Carolina_voters	0.07	548.27	0.0

Table 4.15: inter-cluster similarity β distribution for North-Carolina

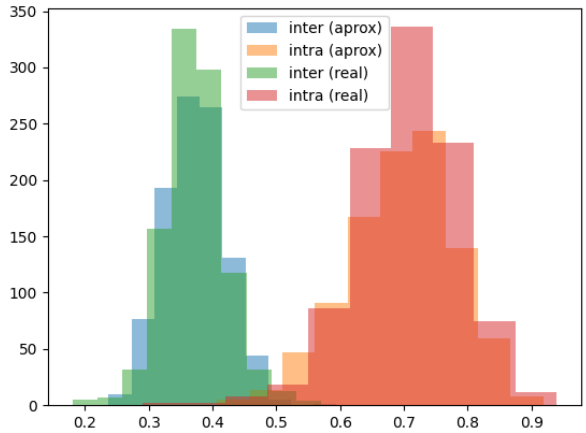


Figure 4.5: A histogram of the approximated β-distributions for the inter- and intra-cluster similarities for the *North-Carolina-Voters* dataset. The β-distribution is scaled to the range between $[-1, 1]$, and the sampled distribution is also plotted for reference.

4.2.3 Artificial data

Data generation

Given the properties of the real-world data sets, artificial datasets have been generated with similar properties. The generated datasets vary in size, the cluster size distributions,and the inter- and intra-cluster similarities.

Data-set sizes The sizes of the datasets can be found in table 4.16. The sizes vary primarily in order of magnitude, in order to be able to judge the performance of the heuristics as dataset size varies.

Number of entities	50	500	1000	10000

Table 4.16: Number of entities per dataset

Discrete cluster size distributions The real-world datasets have cluster sizes that lie primarily within the range $[1, 4]$. Therefore the generated datasets have cluster sizes which lie within the same range. The two used discrete probability distributions can be found in table 4.17 and 4.18. These two distributions represent clusterings which follow the rule: the larger the size of the cluster, the smaller the probability it exists, and vice versa.

Cluster size	1	2	3	4
Probability	0.75	0.15	0.06	0.04

Table 4.17: Descending cluster size distribution

Cluster size	1	2	3	4
Probability	0.04	0.06	0.15	0.75

Table 4.18: Ascending cluster size distribution

Zipf cluster size distribution One more distribution that is used to generate the artificial data is based on Zipf's law. The law models a rank-frequency distribution that is commonly found in data representing the frequency of words within a text-corpus. Given N cluster-sizes, and the shape parameter $s = 1$, then the most frequent cluster-size will occur twice as often as the second most frequent entity, and three times as often as the third most frequently occurring entity. Defined more formally as follows for $k = 1, ..., N$:

$$f(k; s, N) = \frac{1/k^s}{\sum_{n=1}^{N} (1/n^s)}$$

A Zipf distribution with parameters $N = 8$ and $s = 1$ has been used for the artificial data generation.

Similarity distributions The inter- and intra-cluster similarities for the generated datasets both follow a Beta distribution. The distributions used for the generation, described in terms of α and β shape parameters can be found in table 4.19 and 4.20 for the intra- and inter-cluster similarity distributions respectively. The distribution parameters have been chosen so that the amount of overlap between the distributions varies. The reason behind the variation of the amount of overlap of the distribution is because similarity measures can be imperfect. A similarity measure should ideally return a high score for duplicates, and a lower score for non-duplicates. In practice it can happen however that a similarity measure assigns the same score to a pair of duplicates, and a pair of non duplicates. The variation in this overlap serves as a means to evaluate the performance of the heuristics in circumstances in which the discernibility of pairs of entities is less straightforward.

dataset	***alpha***	***beta***	**E(X)**	**SD(X)**
Distribution 1	36	6	0.86	0.05
Distribution 2	27	4	0.87	0.06
Distribution 3	9	1	0.9	0.09

Table 4.19: intra-cluster β distribution parameters

dataset	***alpha***	***beta***	**E(X)**	**SD(X)**
Distribution 1	21	24	0.47	0.07
Distribution 2	14	16	0.47	0.09
Distribution 3	7	8	0.47	0.12

Table 4.20: inter-cluster β distribution parameters

4.3 Experimental setup

4.3.1 Small data-set experiments

Relatively large datasets cannot be solved within reasonable running times with ILP approach since its run time complexity is cubic. Relatively small datasets however can be solved by both the heuristics and ILP approach in reasonable running times. Therefore the datasets of size 50 will be solved with both the heuristics and ILP. Datasets of such size tend not be the representative of real-world data in terms of size. However, the ILP does provide an optimal solution for the costs. Therefore it makes sense to compare its performance with that of the heuristics in order to determine how far the solutions of the heuristic algorithms are removed from the optimal solution.

4.3.2 Large data-set experiments

The remainder of the datasets will be solved in a different manner since the complexity of the problem doesn't allow for the entire dataset to be solved within practical running times. This is because of the complexities of the different heuristics, and the fact that the calculation of all pairwise similarities is a quadratic task. A blocking mechanism is employed so that the heuristics only solve 'blocks' of the data rather than the entire dataset.

4.3.2.1 KNN as a blocking-mechanism

Entity deduplication only becomes a practical task if it can be done without having to compare all pairs, i.e. through a blocking mechanism. A blocking mechanism divides the dataset into 'blocks'. Pairwise comparisons are only performed for entities within the same block [3]. This allows for the dataset to be deduplicated without having to exhaustively calculate the similarity of all pairs .

The employed blocking mechanism uses an approximate k nearest neighbor search for each node based on euclidean distance. An approximate nearest neighbor search can be performed in much less than quadratic time, which makes it faster than an approach in which all pairwise similarities are calculated. Given a dataset D with N nodes, for each node $n \in D$, the pairwise similarity is assessed between n and each of its respective k nearest neighbors. Each pair (n, k_i) which has a similarity greater than θ is placed in the graph G_B. Here θ is a user-defined similarity cut-off. The individual connected components present in G_B form the 'blocks' which are to be solved by the heuristics. Note that although the individual connected components are solved, quality metrics are calculated on the entire dataset.

4.3.3 Determining a suitable cut-off θ

Given a dataset, the cut-off θ that is used defines the problem that is to be solved. The most ideal value for θ would be the one which results in the solutions with the highest F_1 scores. However as was previously mentioned, the required ground-truth to calculate such metrics is often incomplete or entirely unavailable. Therefore an experiment is conducted to understand which θ value leads to optimal F_1 performance. Given a dataset, the heuristics are tuned for a range of θ values $t \in [0, 1]$. For the resulting solutions the F_1 score and costs are calculated.

4.3.4 Hyper-parameter tuning

4.3.4.1 Heuristic parameters

The outlined heuristics have user-defined parameters which influence the performance of the heuristics (i.e. the costs of the output clustering). The respective parameters per heuristic are listed in table 4.21.

Heuristic	**Parameters**
Layout	p_0,f_{att}, M_0, f_{rep}, R, δ_{init}, f_σ, σ
SplitMerge	t_s, t_m
Vote/BOEM	k_{max}
Spectral	k

Table 4.21: Hyper-parameters per heuristic

4.3.4.2 Genetic hyper-parameter tuning

The hyper-parameters are tuned through a genetic tuning process for which the pseudo code can be found in algorithm 3.

Algorithm 3 Genetic hyper-parameter tuning

```
1:  procedure TUNING(heuristic,dataset)
2:
3:      LHSParents ← GenerateParentsLHS(heuristic)
4:      RandomParents ← GenerateParentsRandom(heuristic)
5:      Parents ← RandomParents ∪ LHSParents
6:      Results ← []
7:
8:      CC ← GetConnectedComponents(dataset)
9:      for all i ∈ Parents do
10:         c_i ← 0
11:         for all cc ∈ CC do
12:             c_i ← c_i + EvaluateCosts(cc)
13:         Results.Add((i, c_i))
14:     Results ← LowestScoringTen(Results)
15:
16:     for r = 1, ..., R do
17:         P1 ← GenerateGroupOneParents(heuristic, Results)
18:         P2 ← GenerateGroupTwoParents(heuristic, Results)
19:         P3 ← GenerateGroupThreeParents(heuristic, Results)
20:         Parents ← P1 ∪ P2 ∪ P3
21:
22:         for all i ∈ Parents do
23:             c_i ← 0
24:             for all cc ∈ CC do
25:                 c_i ← c_i + EvaluateCosts(cc)
26:             Results.Add((i, c_i))
27:         Results ← LowestScoringTen(Results)
28:
29:     return Results
```

Lines 1-6 On the third and fourth line of code random parameter configurations along side of Latin-hypercube sampled parameter configurations are generated.

Lines 8-13 Given the dataset on which the heuristic is to be applied, the configurations are applied to randomly picked connected components from the set of the connected components.

Line 14 For each solution the costs are calculated. The configurations are then sorted on aggregate costs. The ten lowest scoring solutions become the parent set for the next step.

Lines 16 - 20 In the next step three sets of 5 parameter configurations are generated. The first set consists of configurations made up of random combinations of parameters values from configurations in the parent set. The second set consists of configurations that are made up of random parameter values. The third set consists of configurations that are a combination of completely random parameter values and randomly picked parameter values from the parent set.

Lines 22-29 The 10 best scoring parameter configurations become the new parent set. This process is repeated for R rounds. The best scoring parameter configuration from the last round is chosen as the resulting tuned configuration.

4.4 Evaluation

4.4.1 Performance measures

A useful clustering should score well on quality measures such as precision and recall. Their formal definitions can be found below, expressed in terms of the true and false positives and negatives. Table 4.22 shows what true-, and false-positives and negatives mean in the context of these experiments.

	Duplicates	Non-duplicates
Same cluster	TP	FP
Different cluster	FN	TN

Table 4.22: Cross table defining true and false positives and negatives

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

The quality measure that combines both measures into one is the F_1 score which is defined as follows:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The F_β-score allows for the importance of precision and recall to be varied through the β parameter. It is defined as follows:

$$F_\beta = \frac{(1 + \beta^2) \cdot (precision \cdot recall)}{\beta^2 \cdot precision + recall}$$

The $F\frac{1}{2}$-score , which weighs precision twice as heavy as recall is defined as the F_β-score with $\beta = 0.5$.

4.4.2 Relation between Costs and F_1-score

Relation within top-k solutions The relation between the costs and F_1-score is investigated for the top-10, top-20 and the entire set of pairs. A dataset is solved with different parameter configurations. The resulting costs and F_1-scores are analysed to determine the relation between the two. This is done for each heuristic, and multiple datasets. In case of the presence of a relation, it allows for the use of the objective cost function as a proxy for the F_1-score. This is useful since it avoid reliance on the ground-truth to assess the performance of the heuristics.

4.4.2.1 Spearman rank correlation coefficient

The relation is analysed using Spearman's rank correlation coefficient. The metric captures how well the relationship between two variables can be represented as a monotonic function given the rankings of these variables. The coefficient is defined as follows:

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)}$$

where d_i is the difference between the two ranks of the i^{th} observation and n is the number of observations. The coefficient lies within the interval $[-1, 1]$ where the sign of the coefficient indicates whether the correlation is either ascending or descending. An associated p-value is also calculated for the hypothesis test in which the null hypothesis is that the two sets of data are uncorrelated.

Chapter 5

Results

The following chapter provides details of the results for the experiments that have been performed. We first present the results of the experiments on the real-world datasets and artificially generated datasets. This is followed by the results for the experiments regarding the rank-correlation between the F_1-scores and costs. Lastly, the results of the experiments in which the θ cut-off value is varied are discussed.

5.1 Results on real data

The experiments were performed on the connected components which result from the *knn* blocking mechanism. Table 5.1 shows the results of the experiments for the real-world data-sets per heuristic. The performance of the heuristics is captured through the F_1-score, the $F\frac{1}{2}$-score and the objective function costs of the solution. The blocking mechanism was performed for $k = 2$ nearest neighbors. The θ cut-off for each data-set can be found in table 5.1, and 5.2. The reported θ-values are those that have resulted in the highest average F_1-score across all heuristics.

In [10] it is stated that for the use of a data-set in systems such as a SPARQL reasoner, precision is more important than recall. A low precision may return irrelevant results as false positives. Therefore the $F\frac{1}{2}$-score is included in order to capture the emphasis on the importance of precision. The $F\frac{1}{2}$-score weighs precision twice as heavy as recall.

	Golden-Agents			Geo			Affiliation		
	F1	$F\frac{1}{2}$	Costs	F1	$F\frac{1}{2}$	Costs	F1	$F\frac{1}{2}$	Costs
Force	**0.84**	0.78	**11.67**	0.75	**0.84**	614.50	0.19	0.37	**366.85**
Split-Merge	0.79	0.71	24.28	**0.77**	0.79	696.53	0.10	0.20	389.89
Spectral	**0.84**	**0.80**	105.11	0.73	0.83	**225.906**	0.13	0.26	402.62
Vote/Boem	0.79	0.73	23.77	0.76	0.79	688.17	0.35	0.50	670.42
Pivot	0.79	0.70	15.70	0.72	0.81	481.50	0.21	0.38	270.58
Closure	0.75	0.67	34.25	0.74	0.75	726.66	**0.38**	**0.51**	803.42
θ	0.85			0.25			0.5		

Table 5.1: Performance measures per real-world data-set per heuristic

	Music			Carolina		
	F1	$F\frac{1}{2}$	Costs	F1	$F\frac{1}{2}$	Costs
Force	0.53	0.45	**226.91**	0.65	0.56	**93.40**
Split-Merge	0.31	0.22	591.91	0.41	0.30	223.92
Spectral	**0.60**	**0.53**	1046.93	**0.70**	**0.62**	540.97
Vote/Boem	0.49	0.39	257.96	0.59	0.48	113.85
Pivot	0.48	0.38	321.74	0.39	0.29	316.34
Closure	0.20	0.13	2179	0.37	0.27	410.96
θ	0.7			0.75		

Table 5.2: (Continued) Performance measures per real-world data-set per heuristic

We observe that results vary per data-set. First we note that the F_1-scores are much lower for the *Affiliation-strings* data-set.

In addition, the*Closure* heuristic performs better than the other heuristics for the *Affiliation-strings* data-set with an F_1-score of 0.38. However, the heuristic is one of the the worst performing for the other data-sets. The *Vote/BOEM* heuristics steadily obtains F_1-scores which are somewhere in between the minima and maxima of the F_1-scores for each data-set.

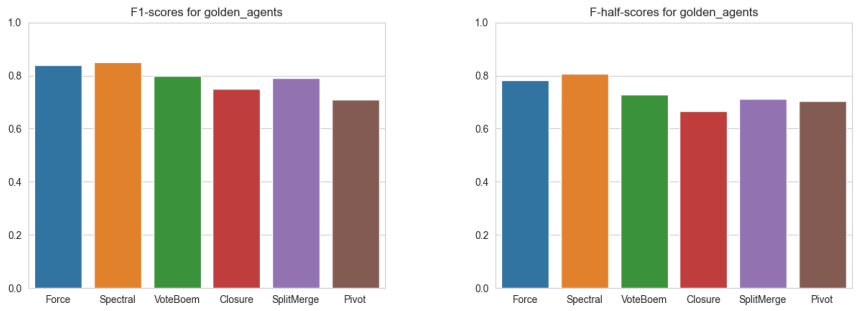
The best performers in terms of the F_1-score for the other data-sets are split between the *Spectral*, *Force*, and *SplitMerge* heuristics. For the *Golden-Agents* data-set, *Force* and *Spectral* share the best performance with an F_1-score of 0.84. Whereas for the *Geographical-settlements* data-set the *SplitMerge* heuristic performs best with an F_1-score of 0.77. The somewhat larger data-sets of *Music-Brainz* and *North-Carolina voters* have been solved best in terms of the F-scores by the *Spectral* heuristic. *Force* performed second best for these two data-sets.

When taking a look at the $F\frac{1}{2}$-scores it is apparent that *Force* and *Spectral* perform better than the other heuristics with the exception of the results for the *Affiliation-strings* data-set, in which they are outperformed by all heuristics but *SplitMerge*. This does indicate that in general *Force* and *Spectral* would generate better results in situation in which precision is more valuable than recall.

With respect to costs, *Force* is consistently one of the best-performing heuristics with a single case of being the second best for the *Geographical-settlements* data-set. In that case *Spectral* performs better.

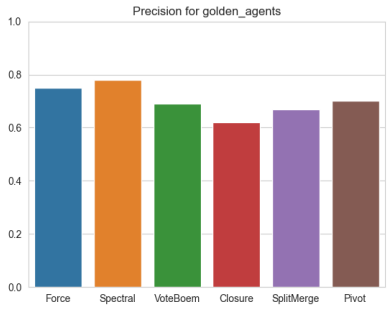
Golden agents

Figure 5.1 shows that with respect to the F_1- and $F\frac{1}{2}$-score *Spectral* and *Force* outperform the other heuristics. *Split-Merge* and *Vote/BOEM* perform only slightly better than *Pivot* and *Closure*. What is interesting is that the heuristics perform relatively similar for recall. The primary cause of the differences in F_1- and $F\frac{1}{2}$-scores between the heuristics is due to their precision.

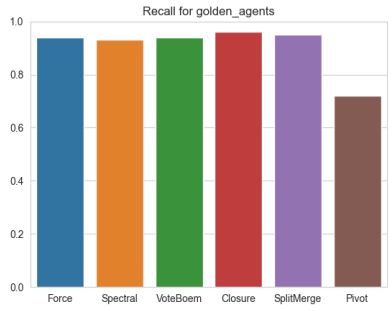


(a) F1

(b) F$\frac{1}{2}$



(c) Precision

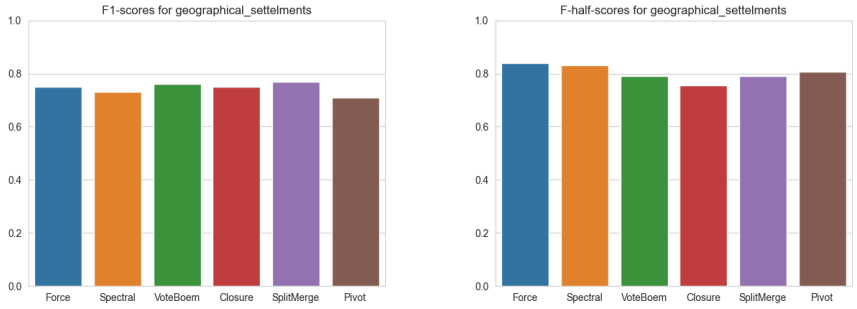


(d) Recall

Figure 5.1: Performance of the clustering algorithms on the Golden Agents data.

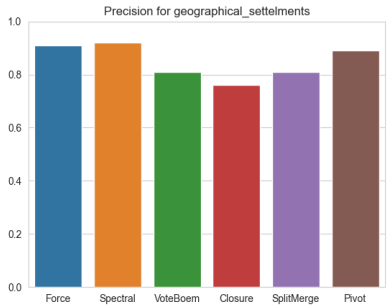
Geographical settlements

Figure 5.2 shows that in terms of F_1-scores, the performance of the different heuristics is fairly similar. However, when taking a look at the other subfigures in figure 5.2, differences in performance do appear. *Force* and *Spectral* outperform the other heuristics in terms of $F\frac{1}{2}$-score. This is once again the case due to the fact that the two score somewhat higher in terms of precision in comparison to the other heuristics. Interestingly enough, the other heuristics do however have a higher recall.

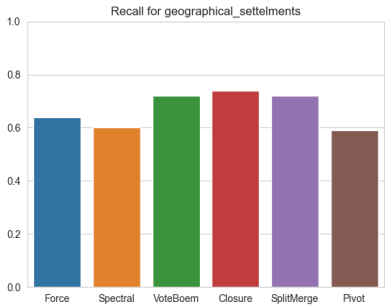


(a) F1

(b) F$\frac{1}{2}$



(c) Precision



(d) Recall

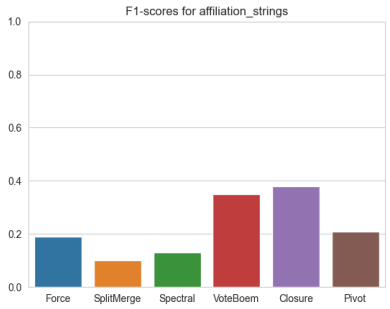
Figure 5.2: Performance of the clustering algorithms on the Geographical settlements data.

Affiliation strings

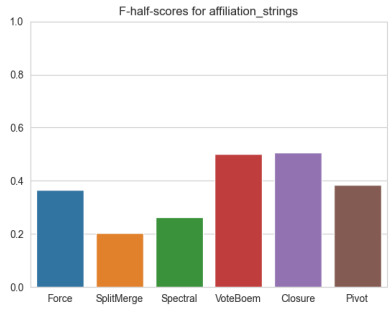
The results for the *Affiliation-strings* data-set differ from the trend that we've seen before. Figure 5.3 shows that the clear winners in terms of both the *F1*- and $F\frac{1}{2}$-scores are not *Force* and *Spectral*, but rather *Closure*, and *Vote/BOEM*.

This can be most likely attributed to the *knn* process, and the fact that $k = 2$ leads to relatively small connected components.Primarily because for the *Affiliation-strings* dataset the ground truth clusters are exceptionally large for an entity deduplication data set. This favours solutions which return the transitive closure of the connected component since it is the case that most entities within these connected components are duplicates. However it is interesting that even in such cases *Spectral*, *SplitMerge*, and *Force* produce solutions with higher precision.

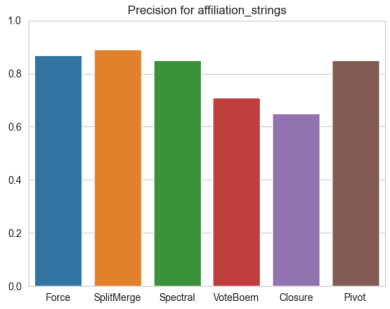
What is interesting however is that in terms of precision *Spectral*, *SplitMerge*, and *Force* perform better than *Pivot*, *Closure*, and *Vote/BOEM*.



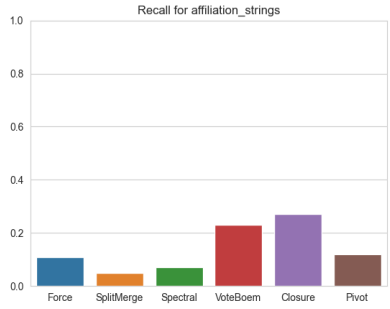
(a) F1



(b) $F\frac{1}{2}$



(c) Precision

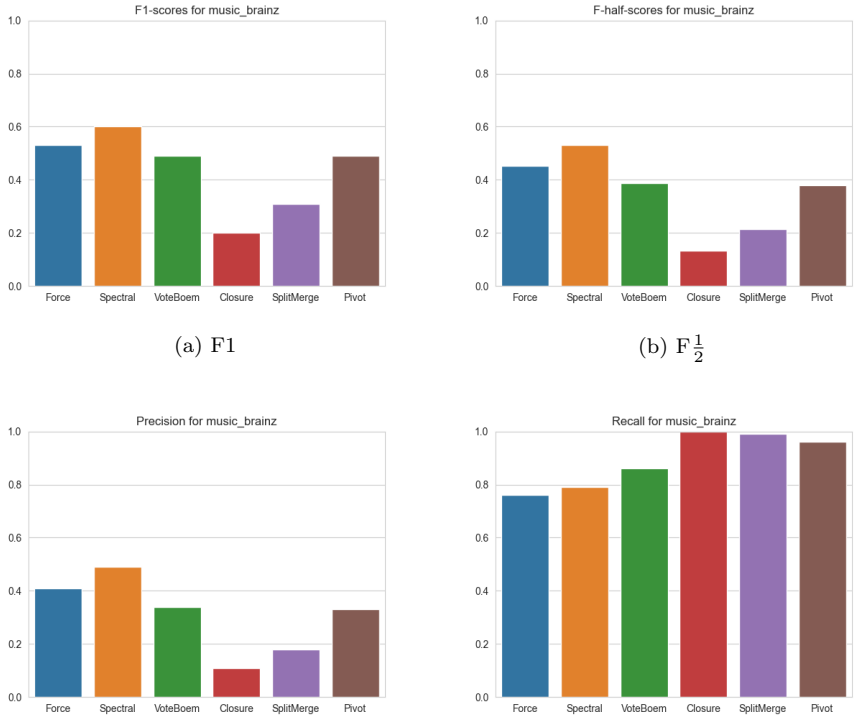


(d) Recall

Figure 5.3: Performance of the clustering algorithms on the Affiliation strings data.

Music-Brainz

The results for *Music-Brainz* can be found in figure 5.4. A similar trend holds for this data-set. *Closure* is outperformed by the other heuristics. In this case it is outperformed for both the F_1-score and the $F\frac{1}{2}$-score. The precision stats show that *Pivot* and *Closure* perform much worse than the others, explaining the differences in F-scores. *Split-Merge* performs better than *Closure*, though worse than the others. *Split-Merge* does however obtain a better recall-score together with *Pivot*.



(a) F1

(b) F$\frac{1}{2}$

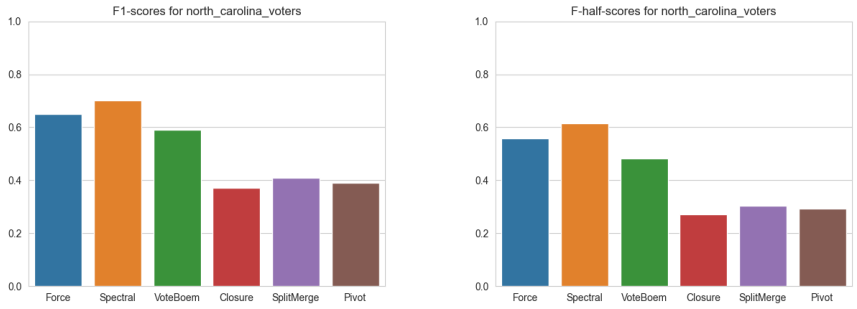
(c) Precision

(d) Recall

Figure 5.4: Performance of the clustering algorithms on the Music-Brainz data.

North-Carolina voters

Figure 5.5 shows that similar trends emerge as before. *Force*, *Vote/BOEM* and *Spectral* perform best in terms of F-scores. *Split-Merge* performs slightly better than *Pivot* and *Closure*. The differences in scores for precision and recall also follow similar trends as for the majority of the other data-sets. *Force*, *Vote/BOEM* and *Spectral* outperform the other heuristics when it comes to precision-scores.



(a) F1

(b) $F\frac{1}{2}$

(c) Precision

(d) Recall

Figure 5.5: Performance of the clustering algorithms on the North-Carolina voters data.

5.2 Results on artificial data

Figures 5.6, 5.7, 5.8, and 5.9 show an overview of average F_1, precision and recall scores per dataset-size. The averages are taken over all results across all theta-values for all datasets of the same size. The averages are taken for the datasets that have been made with different cluster-size-distributions and similarity-distributions. The datasets of size 50 are small enough for the ILP to solve them within reasonable time, hence for these datasets the results of the ILP approach are included as well. The full results can be found in Appendix A.

The knn blocking mechanism that was used for the real-world data-sets serves as a means to avoid computing all pairwise-similarities. The artificial data-sets contain pre-computed pairwise similarities, which removes the need for the knn process since the similarities are readily available. Therefore, for the artificial data-sets all connected components which consist of edges with a weight larger than θ are solved.

5.2.1 Datasets of size 50

Figure 5.6 shows that the heuristics can be split into one of two performance categories. Namely those which perform better in terms of precision than recall (*Force*, *Spectral*, *Vote/BOEM*, *Pivot*, and *Split-Merge*), and vice versa (*Closure*). What is interesting is that and *Closure* outperforms the rest significantly in terms of recall. However, it must also be noted that it is outperformed by all other heuristics in terms of precision. In this case for the heuristics *Force* is the best performer in terms of precision, followed by *Vote/BOEM*. In Appendix A it can be seen that the heuristics tend to score highest on F_1-scores for the datasets generated with similarity distribution 1, followed by distribution 2, which is followed by distribution 3.

The *ILP* performs best in terms of F_1, and precision. In terms of recall it is outperformed by *Closure*.

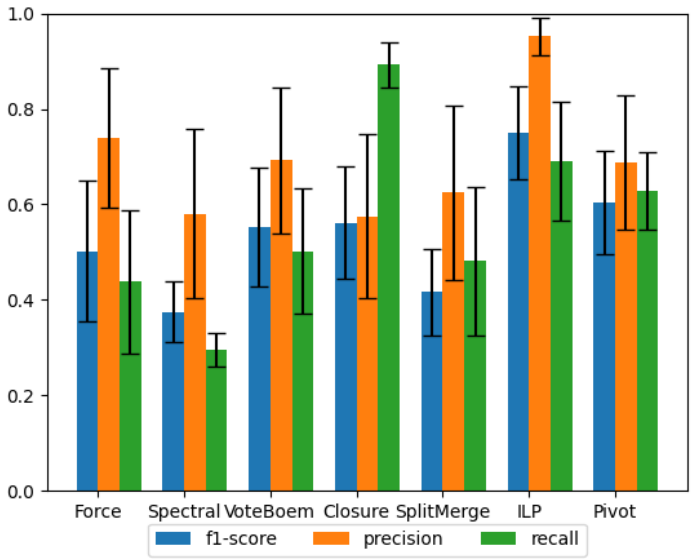


Figure 5.6: F_1, precision and recall per heuristic for data-sets of size 50. The error bars represent the variance.

5.2.2 Datasets of size 500

Figure 5.7 shows that the trend in which *Closure* performs better for recall than precision barely holds for the data-sets of size 500. What is interesting however is that as the data-set size grows, *Force*, *Spectral*, and *Vote/BOEM* perform better than *Pivot* and *Closure* in terms of precision. The two best performing heuristics in terms of precision are, just like for the data-sets of size 50, [Vote/BOEM] and *Force*. In Appendix A it can be seen that the heuristics tend to score highest on F_1-scores for the datasets generated with similarity distribution 1, followed by distribution 2, which is followed by distribution 3.

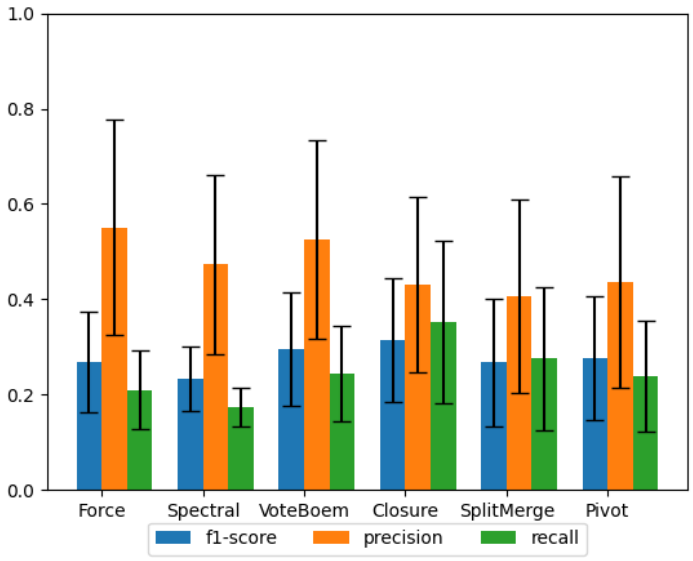


Figure 5.7: F_1, precision and recall per heuristic for data-sets of size 500. The error bars represent the variance.

5.2.3 Datasets of size 1000

The bar-plots in figure 5.8 show that as the data-set size grows, *Force*, *Spectral*, *Vote/BOEM* and *Split-Merge* perform better than *Pivot* and *Closure* in terms of precision. The two best performing heuristics in terms of precision are in this case *Force*, and *Vote/BOEM*. It must be noted that as the data-sets have grown the recall in general has decreased across all solutions generated by the heuristics. In Appendix A it can be seen that the heuristics tend to score highest on F_1-scores for the datasets generated with similarity distribution 1, followed by distribution 2, which is followed by distribution 3.

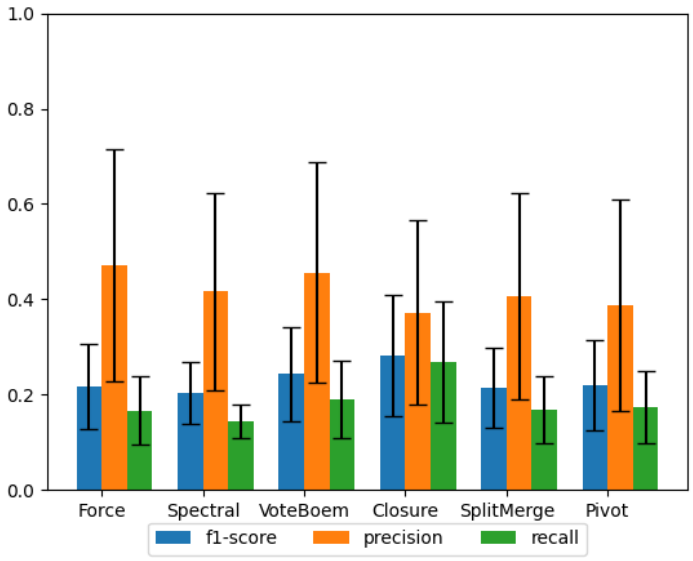


Figure 5.8: F_1, precision and recall per heuristic for data-sets of size 1000. The error bars represent the variance.

5.2.4 Datasets of size 10.000

The results for the data-sets of size 10.000 in figure 5.9 show that the gap in terms of precision between the different heuristics narrows. On top of that does the recall drop as the data-set size increases for all heuristics. What is noticeable however is that the trend of *Pivot* and *Closure* being outperformed in terms of precision by the other heuristics besides *Split-Merge* still remains the case. In this case *Vote/BOEM* performs best in terms of precision, though it is closely followed by *Force* and *Spectral*. In Appendix A it can be seen that the heuristics tend to score highest on F_1-scores for the datasets generated with similarity distribution 1, followed by distribution 2, which is followed by distribution 3. Since this trend was present for all sizes for the generated datasets, the case can be made that as the overlap between similarity distributions increases the maximum F_1-score decreases.

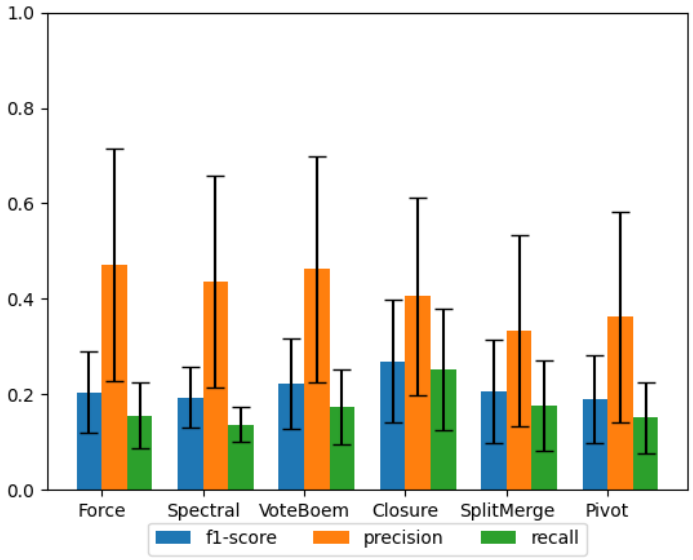


Figure 5.9: F_1, precision and recall per heuristic for data-sets of size 10.000. The error bars represent the variance.

5.3 Rank-correlation results

In this section the costs and F_1-scores are calculated for solutions that have been generated with different hyper-parameters. This is done for the *Force SplitMerge*, and *Vote/BOEM* heuristics since they are the only heuristics which have user-defined parameters.

Per data-set the Spearman rank correlation coefficient is calculated for both the top-10 and top-30 cost F_1-pairs. The correlation indicates the degree of the correlation and whether it is a positive or negative correlation. The p-value indicates the significance of the correlation.

5.3.1 Force

In table 5.3 the Spearman rank correlation coefficient can be found for the *Force* heuristic for the *Golden-Agents*, *Geographical-settlements* and the *Affiliation-strings* data-sets.

	Top-	*correlation*	*p-value*
Golden-Agents	10	0.44	0.209
	30	-0.86	1.01×10^{-9}
Geographical	10	0.20	0.57
	30	-0.73	5.05×10^{-6}
Affiliation	10	0.46	0.18
	30	-0.62	2.3×10^{-5}

Table 5.3: Spearman rank correlation for Force

The correlation coefficient for the top-30 best costs of all three data-sets are negative with an associated p-value smaller than 0.05. The *Golden-Agents*, *Geographical-settlements* and *Affiliation-strings* have correlation coefficients of -0.86, -0.73 and -0.62 respectively. Hence making it safe to assume that due to their significance levels that there is a negative correlation between the costs and F_1-scores for the top-30 best performing solutions. The coefficients do vary in their values, and hence the correlation of the relation as well. This goes to show that the correlation can be expected to be negative, though it can vary as data-set properties vary as well.

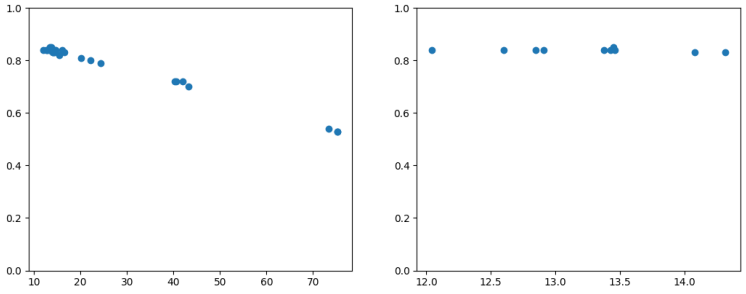


Figure 5.10: Costs vs F_1-scores for *Golden-Agents* using the *Force* heuristic for the top-30 (left) and top-10 (right)

	Top-	*Mean*		*Variance*	
		Costs	F_1	Costs	F_1
Golden-Agents	10	13.25	0.84	0.41	2.9×10^{-5}
	30	24.56	0.79	365.31	0.01
Geographical	10	597	0.68	8.31	6.4×10^{-5}
	30	655.44	0.62	7455.29	0.01
Affiliation	10	364.31	0.17	11.52	4.9×10^{-5}
	30	378.01	0.163	286.08	0.0002

Table 5.4: Mean and variances for costs and F_1 for *Force*

For the top-10 best performing solutions the rank-correlation coefficient follows a different trend. Namely that the coefficients are positive. Though note that the coefficients are much closer to zero with values of 0.44, 0.20, and 0.46. On top of that, the respective p-values are much larger than 0.05, indicating that the correlation is relatively insignificant. This is in part caused by the small sample size of 10. Even though there is a negative correlation between the top-30 pairs of costs and F_1-scores, it is not monotonically related for the top-10. This is visually shown as well in figure 5.10

Table 5.4 shows both the mean and variance in costs and F_1-scores for the top-10 and top-30 for the different data-sets. The table shows that for the top-10 the mean costs are lower and mean F_1 higher than for the top-30. The variance for both the costs and F_1 are smaller for the top-10 than the top-30.

5.3.2 Split-Merge

In table 5.5 the Spearman rank correlation coefficient can be found for the *Split-Merge* heuristic for the *Golden-Agents*, *Geographical-settlements* and the *Affiliation-strings* data-sets.

	Top-	*correlation*	*p-value*
Golden-Agents	10	-0.1	0.79
	30	-0.91	4.90×10^{-12}
Geographical	10	-0.1	0.77
	30	-0.79	1.7×10^{-7}
Affiliation	10	0.02	0.96
	30	0.79	1.93×10^{-7}

Table 5.5: Spearman rank correlation for Split-Merge

The correlation coefficient for the top-30 best costs for two of the three data-sets are negative with an associated p-value smaller than 0.05. The *Golden-Agents*, *Geographical-settlements* and *Affiliation-strings* have correlation coefficients of -0.91, -0.79 and 0.79 respectively. Hence making it safe to assume that due to their significance levels that there is a negative correlation between the costs and F_1-scores for the top-30 best performing solutions for *Golden-Agents* and *Geographical-settlements*, and a positive correlation for *Affiliation-strings*. This goes to show that it is not necessarily to be expected that the relation-ship between costs and the F_1-score is a negative one.

	Top-	*Mean*		*Variance*	
		Costs	F_1	Costs	F_1
Golden-Agents	10	25.71	0.81	1.09	0.0004
	30	47.36	0.65	1325.89	0.084
Geographical	10	672.08	0.77	669.83	0.0002
	30	728.46	0.71	4154.53	0.01
Affiliation	10	421.71	0.13	3702.17	0.005
	30	647.32	0.27	30510.7	0.02

Table 5.6: Mean and variances for costs and F_1 for *Split-Merge*

For the top-10 best performing solutions the rank-correlation coefficient follows a different trend. Note that the coefficients are much closer to zero with values of -0.1, -0.1, and 0.02. The associated p-values also indicate that the correlation is insignificant for the top-10. This does go to show that whatever trend that is present in the top-30 of cost and F_1 pairs, dissipates as the solutions become better in terms of costs. Table 5.6 shows the same trend as table 5.4.

5.3.3 Vote/BOEM

In table 5.7 the Spearman rank correlation coefficient can be found for the *Vote/BOEM* heuristic for the *Golden-Agents*, *Geographical-settlements* and the *Affiliation-strings* data-sets.

	Top-	*correlation*	*p-value*
Golden-Agents	10	1	0.0
	30	-0.4	0.027
Geographical	10	0.17	0.63
	30	-0.035	0.85
Affiliation	10	0.16	0.64
	30	0.71	1.2×10^{5}

Table 5.7: Spearman rank correlation for Vote/BOEM

	Top-	*Mean*		*Variance*	
		Costs	F_1	Costs	F_1
Golden-Agents	10	23.24	0.8	0.02	0
	30	23.78	0.79	1.01	9.8×10^{-6}
Geographical	10	684.18	0.76	0.33	9×10^{-6}
	30	688.91	0.75	27.57	2.1×10^{-5}
Affiliation	10	629.94	0.33	49.42	9×10^{-6}
	30	651.34	0.34	487.78	6.5×10^{-5}

Table 5.8: Mean and variances for costs and F_1 for *Vote/BOEM*

Interestingly, for *Vote/BOEM* the correlation coefficient for the top-30 for *Geographical-settlements* is not significant with a p-value of 0.85. The coefficients of -0.4 and 0.71 for the top-30 of *Golden-Agents* and *Affiliation-strings* respectively are significant with a p-value lower than 0.05. None of the coefficients for the top-10's are significant however, except for *Golden Agents*

One possible explanation for the lack of correlation for the *Geographical-settlements* data-set could be explained by the *BOEM* process. Namely in that it performs the *'best one element move'*. Given that one of the hyper-parameters for *Vote/BOEM* is the maximum number of iterations for *BOEM*, it could be that all parameters that have been used to generate the solutions for *Geographical-settlements* result in the same number of one-element-moves to have been performed. Therefore explaining the similarity in score for the data-set. Table 5.8 shows the same trend as table 5.4.

5.4 Rank-Correlation across heuristics

All in all,for the top-10 best performing solutions for *Force*, *Split-Merge*, *Vote/BOEM* there is not a clear correlation between costs and F_1-score across different data-sets. For the top-30 best solutions there isn't a rather clear correlation between costs and F_1-score for the different heuristics in most cases. It is noticeable however that for the *Affiliation-strings* data-set the correlation tends to be positive. This can be explained by the size of the cluster in the data-set. Primarily because the connected components that emerge from the *knn*-process for *Affiliation-strings* can be smaller than the ground-truth cluster. Hence favouring solutions which return the transitive closure of the connected component since it is the case that most entities within these connected components are duplicates.

5.5 θ cut-off variation

In figures 5.11, 5.12, and 5.13 the obtained F_1-, precision- and recall-scores vs. the θ cut-off can be found for the *Golden-Agents*, *Geographical-settlements* and *Affiliation-strings* respectively from left to right.

Golden-Agents

For the *Golden-Agents* data-set it can be seen that for all heuristics that the F_1-score roughly peaks around a θ cut-off of roughly 0.8. The heuristics climb in terms of precision as the θ value increases, and conversely the recall goes down as the θ value increases.

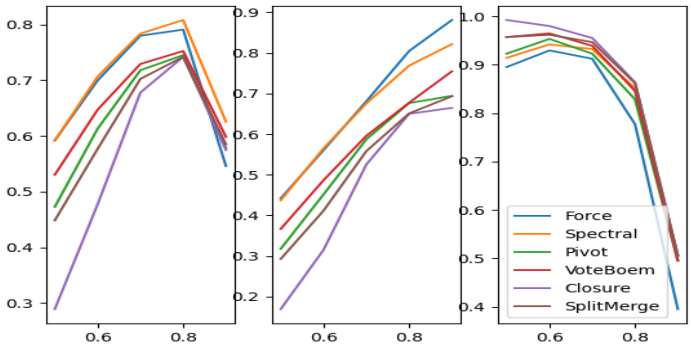


Figure 5.11: F_1, Precision, Recall vs. θ for the different heuristics for the Golden-Agents data.

Geographical-settlements

For the *Geographical-settlements* data-set the heuristics also follow a similar trend to one another for F_1, precision, and recall. The optimal θ cut-off value for the F_1-score lies around 0.25, and drops as θ increases. Here too the precision increases roughly as θ increases, and recall decreases as θ approaches one.

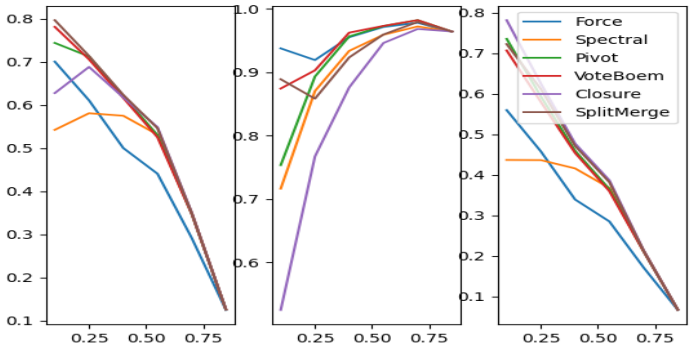


Figure 5.12: F1, Precision, Recall vs. θ for the different heuristics for the Geographical Settlements data.

Affiliation-Strings

For the *Affiliation-strings* data-set there is more variation between the different heuristics and their performance measures as θ varies. Most heuristics perform best around the 0.5 mark, though here the heuristics vary too. *Pivot*, *Closure*, and *Vote/BOEM* perform better than *Force*, *Spectral*, and *Split-Merge* for the F_1-score and recall. Though they also perform worse in terms of precision. As θ goes up, the trend of the F_1-score going down, the precision going up, and the recall going down is the case just as for the other data-sets.

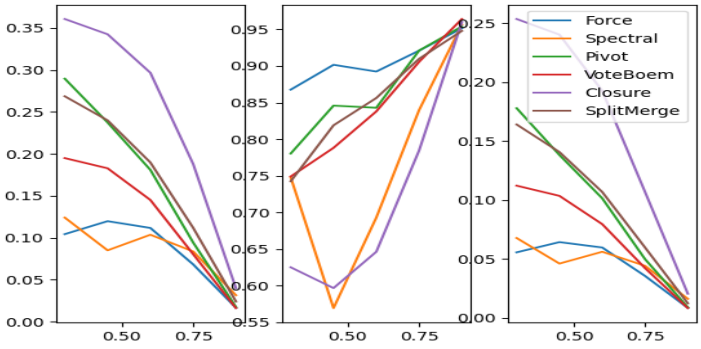


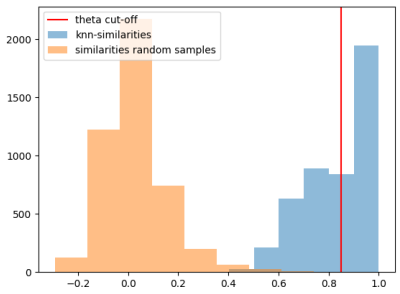
Figure 5.13: F1, Precision, Recall vs. θ for the different heuristics for the Affiliation strings data.

When comparing the results with varying θ cut-off values for the different data-sets it is apparent that the results are different for the *Affiliation-Strings* data-set. Namely in that *Pivot*, *Closure* outperform the other heuristics. This once again can be most likely attributed to the *knn* process. Primarily because the connected components that emerge from the *knn*-process for *Affiliation-strings* can be smaller than the ground-truth cluster. Hence favouring solutions which return the transitive closure of the connected component since it is the case that most entities within these connected components are duplicates.

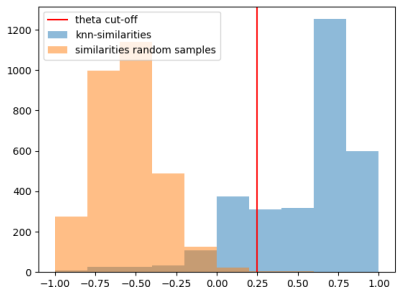
However it must be noted that *Force*, *Spectral*, and *Split-Merge* perform better in terms of precision. This makes the case that these are better candidates for entity-deduplication in scenarios where precision is valued more than recall.

5.6 θ cut-off results

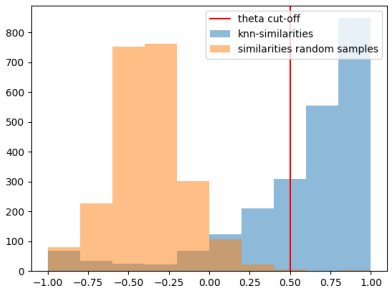
Given a cut-off θ, what is quite interesting is that the optimal θ-value with respect to the F_1-score for each data-set shares a similar property. Namely that the empirical distribution of similarities of randomly sampled pairs of entities, then roughly 99% of these similarities are smaller than the optimal θ-value. This makes the distribution similarities of randomly sampled pairs interesting with regards to picking a suitable θ cut-off point. This phenomenon can be seen in figure 5.14. The red lines represent the optimal θ values for the respective data-sets. The distribution of similarities for the pairs of entities resulting from a *knn*-search for $k = 2$ are also included in the histograms. It is apparent that a significant part of pairwise similarities for these entities have values higher than θ.



(a) Golden Agents



(b) Geographical Settlements



(c) Affiliation Strings

Figure 5.14: Plot of the similarity distributions of sampled pairs (in orange), and a red line representing the optimal θ cut-off for the F_1-score.

Table 5.9 shows the percentage of similarities for all randomly sampled pairs (i, j) with a similarity value $s(i, j)$ that is smaller or equal to θ. This percentage is higher than 99% for the *Golden agents*, *Geographical settlements*, and the *Affiliation strings* dataset. This shows that it is interesting to use the distribution of similarities for randomly sampled pairs as a heuristic for θ selection. It must be noted that this approach takes advantage of the scarcity of duplicate pairs in these data-sets, which makes it safe to assume that most randomly sampled pairs are not duplicates to begin with.

	Golden Agents	***Geographical***	***Affiliation***
$s(i, j) \leq \theta$	0.999	0.998	0.996

Table 5.9: Ratio of similarities with similarity $s(i, j) \leq \theta$ for all randomly sampled pairs (i, j)

Chapter 6

Conclusion

The research conducted for this thesis served as its main purpose to provide insight the application of heuristic weighted cluster-edit algorithms for entity-deduplication. More specifically, insight into which heuristic weighted cluster-edit algorithms perform best (both within and outside of the scope of entity-deduplication), what the relationship is between their cost-function and the F_1-score, and whether the properties of entity-deduplication data-sets can be typified.

Research question 1.1 *Which heuristic algorithms for weighted cluster-editing perform best?*

The results for the generated artificial data-sets of different sizes show that on average *Force*, *Split-Merge*, *Spectral*, and *Vote/BOEM* perform better than *Pivot* and *Closure* when it comes to precision scores. Making the case that all but *Pivot* and *Closure* show more merit for use in systems in which precision is valued more than recall.

The artificial data results show a trend that as the dataset size increases, the performance decreases. For all sizes for the generated datasets, the case can be made that as the overlap between similarity distributions increases the maximum F_1-score decreases as well.

Research question 1.2 *What is the relation between the value of the objective function and the F_1 score?*

The research has shown that for different heuristics and different data-sets there is a rank correlation between the costs and F_1-score when a broad spectrum of cost-F_1 pairs are assessed. This correlation is however less present when the scope is narrowed down to the top-10 best-performing cost-F_1 pairs for all investigated heuristics and data-sets. For almost all data-sets and heuristics the rank-correlation between costs and F_1-score is negative. This goes to show that there is merit in the tuning of hyper-parameters by using costs.

Research question 2.1 *Which heuristic algorithms for weighted cluster-editing perform best for the entity-deduplication problem?*

The results of the real-world data-sets have shown that which heuristic performs best with respect to the F_1-score can vary per data-set. It is noticeable however that *Force* performs best when it comes to obtaining a solution with low costs. These results have also shown that if the *knn* process produces connected components that are larger than most ground-truth clusters, then *Pivot* and *Closure* are outperformed by *Force*, *Split-Merge*, *Spectral*, and *Vote/BOEM*. On top of that do the results show that *Force*, *Split-Merge*, *Spectral*, and *Vote/BOEM* in general perform better than *Pivot* and *Closure* in terms of precision score, and hence this difference in performance is also reflected in the $F\frac{1}{2}$-scores. This goes to show that the heuristics (excluding *Pivot* and *Closure*) have merit for the use in contexts where precision is deemed more valuable than recall.

Vote/BOEM never performs as the best heuristic for any of the real datasets. Although *Force* is the overall best performing heuristics, there is still a case to be made for *Spectral*. This is because *Force* needs to tune seven hyper-parameters, whereas *Spectral* does not need to be tuned when using the *eigengap* heuristics.

Research question 2.2 *What are the typical properties of entity deduplication problems in terms of the distribution of cluster sizes and weights?*

It has been shown that for the intra- and inter-cluster similarities of entity deduplication datasets common distributions were rejected by the Kolmogorov-Smirov test. The Beta distribution gave the best fit among the distributions considered. Visual histogram comparisons of the modeled- distributions and empirical-distributions do show that despite the rejection, they do resemble one another.

Finally, the results concerning optimal θ cut-off selection for optimal F_1-scores have shown that although the θ values can vary per data-set, it could be that the optimal θ value coincides with the cut-off for which 99.9% of randomly sampled pairwise similarities are smaller than θ. This can indicate that the heuristics perform best when the connected-components that are to be solved are relatively 'noise-free', i.e. the majority of the edges included in these components belong to the ground-truth. Further research could investigate to what extent there is any merit to these observations.

Appendix A

Artificial data results

The tables present in this appendix contain the results for the experiments on the artificially generated data. The results have been grouped in a section per data-set size. Each table contains the results for the artificial data-sets that have been generated with the similarity distributions $1 - 3$ that have been defined in tables 4.19 and 4.20. Dist. 1, Dist. 2 represent the cluster-size distributions that have been defined in tables 4.17 and 4.18 respectively. The tables show per data-set, per heuristics the average and maximum F_1-scores for the results for all θ-values between 0.1 and 0.9 with intervals of 0.1.

A.1 Datasets of size 50

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge		ILP	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.79	0.99	0.54	0.73	0.72	1.0	0.83	1.0	0.73	1.0	0.73	1.0	0.7	0.94
Dist. 2	0.59	1.0	0.33	0.56	0.73	0.96	0.58	1.0	0.59	1.0	0.53	1.0	0.8	1.0
Zipf Dist.	0.8	1.0	0.55	0.69	0.74	0.97	0.81	1.0	0.75	1.0	0.45	0.94	0.77	0.97

Table A.1: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 50 generated with similarity distributions 1

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge		ILP	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.44	1.0	0.38	0.74	0.6	0.89	0.47	1.0	0.46	0.96	0.32	0.77	0.81	0.93
Dist. 2	0.43	0.99	0.33	0.58	0.73	0.98	0.51	1.0	0.55	1.0	0.29	0.71	0.83	0.99
Zipf Dist.	0.42	1.0	0.33	0.71	0.68	0.97	0.49	1.0	0.51	1.0	0.23	0.53	0.81	0.99

Table A.2: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 50 generated with similarity distributions 2

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge		ILP	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.13	0.46	0.15	0.52	0.44	0.77	0.22	0.62	0.26	0.79	0.39	0.52	0.81	0.92
Dist. 2	0.27	0.58	0.19	0.6	0.54	0.8	0.34	0.71	0.36	0.96	0.32	0.43	0.83	0.94
Zipf Dist.	0.16	0.65	0.14	0.58	0.46	0.63	0.24	0.65	0.27	0.89	0.37	0.46	0.84	0.95

Table A.3: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 50 generated with similarity distributions 3

A.2 Datasets of size 500

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.38	0.95	0.33	0.71	0.48	0.96	0.41	0.96	0.36	0.9	0.32	0.9
Dist. 2	0.29	0.92	0.21	0.52	0.33	0.95	0.31	0.94	0.35	0.88	0.2	0.91
Zipf Dist.	0.37	0.97	0.28	0.65	0.5	0.97	0.38	0.99	0.37	0.97	0.3	0.98

Table A.4: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 500 generated with similarity distributions 1

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.22	0.73	0.22	0.56	0.23	0.85	0.27	0.81	0.23	0.69	0.23	0.69
Dist. 2	0.18	0.54	0.16	0.5	0.2	0.71	0.19	0.55	0.26	0.79	0.26	0.79
Zipf Dist.	0.19	0.58	0.18	0.5	0.23	0.9	0.22	0.67	0.22	0.71	0.14	0.7

Table A.5: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 500 generated with similarity distributions 2

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.12	0.59	0.13	0.63	0.11	0.68	0.14	0.7	0.17	0.87	0.16	0.79
Dist. 2	0.12	0.62	0.11	0.54	0.11	0.68	0.14	0.68	0.18	0.89	0.18	0.89
Zipf Dist.	0.13	0.63	0.12	0.61	0.11	0.69	0.14	0.68	0.19	0.93	0.19	0.93

Table A.6: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 500 generated with similarity distributions 3

A.3 Datasets of size 1000

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.32	0.95	0.28	0.74	0.4	0.97	0.35	0.98	0.32	0.91	0.27	0.91
Dist. 2	0.27	0.98	0.21	0.58	0.3	0.97	0.29	0.98	0.35	0.97	0.14	0.37
Zipf Dist.	0.28	0.92	0.24	0.64	0.3	0.97	0.3	0.96	0.32	0.9	0.31	0.9

Table A.7: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 1000 generated with similarity distributions 1

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.15	0.41	0.17	0.54	0.18	0.58	0.18	0.46	0.17	0.65	0.19	0.58
Dist. 2	0.09	0.33	0.11	0.47	0.1	0.44	0.11	0.4	0.18	0.73	0.18	0.73
Zipf Dist.	0.11	0.35	0.13	0.5	0.13	0.49	0.14	0.42	0.2	0.72	0.16	0.54

Table A.8: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 1000 generated with similarity distributions 2

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.13	0.65	0.13	0.66	0.12	0.72	0.14	0.71	0.18	0.89	0.09	0.47
Dist. 2	0.13	0.64	0.11	0.57	0.12	0.7	0.14	0.68	0.19	0.93	0.16	0.82
Zipf Dist.	0.13	0.65	0.12	0.6	0.11	0.68	0.13	0.67	0.18	0.89	0.08	0.4

Table A.9: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 1000 generated with similarity distributions 3

A.4 Datasets of size 10000

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.29	0.96	0.28	0.77	0.33	0.97	0.32	0.99	0.33	0.93	0.24	0.93
Dist. 2	0.25	0.91	0.19	0.54	0.31	0.95	0.26	0.93	0.3	0.86	0.3	0.89
Zipf Dist.	0.25	0.97	0.21	0.64	0.0	0.0	0.27	0.98	0.3	0.95	0.21	0.95

Table A.10: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 10000 generated with similarity distributions 1

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.15	0.37	0.16	0.54	0.17	0.52	0.16	0.44	0.17	0.69	0.07	0.35
Dist. 2	0.09	0.35	0.11	0.47	0.1	0.47	0.11	0.41	0.18	0.75	0.15	0.75
Zipf Dist.	0.11	0.39	0.13	0.5	0.0	0.0	0.12	0.45	0.19	0.73	0.16	0.58

Table A.11: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 10000 generated with similarity distributions 2

	Force		Spectral		Pivot		VoteBoem		Closure		SplitMerge	
	Avg.	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**	**Avg.**	**Max**
Dist. 1	0.12	0.6	0.13	0.64	0.12	0.71	0.14	0.69	0.17	0.85	0.15	0.73
Dist. 2	0.12	0.6	0.11	0.55	0.11	0.66	0.13	0.64	0.18	0.9	0.07	0.34
Zipf Dist.	0.13	0.64	0.12	0.59	0.0	0.0	0.14	0.68	0.18	0.91	0.18	0.91

Table A.12: Average and Max F_1-scores for the results for each heuristic for the artificial datasets of size 10000 generated with similarity distributions 3

Bibliography

[1] In Lee. Big data: Dimensions, evolution, impacts, and challenges. *Business horizons*, 60(3):293–303, 2017.

[2] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. An overview of end-to-end entity resolution for big data. *ACM Computing Surveys (CSUR)*, 53(6):1–42, 2020.

[3] Anish Das Sarma, Ankur Jain, Ashwin Machanavajjhala, and Philip Bohannon. An automatic blocking mechanism for large-scale de-duplication tasks. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1055–1064, 2012.

[4] Sebastian Böcker and Jan Baumbach. Cluster editing. In Paola Bonizzoni, Vasco Brattka, and Benedikt Löwe, editors, *The Nature of Computation. Logic, Algorithms, Applications*, pages 33–44, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[5] Sven Rahmann, Tobias Wittkop, Jan Baumbach, Marcel Martin, Anke Truss, and Sebastian Böcker. Exact and heuristic algorithms for weighted cluster editing. In *Computational Systems Bioinformatics: (Volume 6)*, pages 391–401. World Scientific, 2007.

[6] Jurian Baas, Mehdi Dastani, and Ad Feelders. Exploiting transitivity constraints for entity matching in knowledge graphs. *arXiv preprint arXiv:2104.12589*, 2021.

[7] Alieh Saeedi, Markus Nentwig, Eric Peukert, and Erhard Rahm. Scalable matching and clustering of entities with famer. *Complex Systems Informatics and Modeling Quarterly*, (16):61–83, 2018.

[8] Frank McSherry. Spectral partitioning of random graphs. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 529–537. IEEE, 2001.

[9] Micha Elsner and Warren Schudy. Bounding and comparing methods for correlation clustering beyond ilp. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 19–27, 2009.

[10] Juriaan Baas, Mehdi M Dastani, and Ad J Feelders. Entity matching in digital humanities knowledge graphs. *Proceedings http://ceur-ws. org ISSN*, 1613:0073, 2021.

[11] Gary M Weiss. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter*, 6(1):7–19, 2004.

[12] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992.

[13] Wanzeng Kong, Changsihe Sun, Sanqing Hu, and Jianhai Zhang. Automatic spectral clustering and its application. In *2010 International Conference on Intelligent Computation Technology and Automation*, volume 1, pages 841–845. IEEE, 2010.

[14] Anke Van Zuylen and David P Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. *Mathematics of Operations Research*, 34(3):594–620, 2009.

[15] Jurian Baas, Mehdi Dastani, and Ad Feelders. Tailored graph embeddings for entity alignment on historical data. In *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services*, pages 125–133, 2020.

[16] Alieh Saeedi, Eric Peukert, and Erhard Rahm. Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In *European Conference on Advances in Databases and Information Systems*, pages 278–293. Springer, 2017.