Leveraging Clustering Algorithms on Connected Components for Entity Resolution

Master thesis

Frank Hoogmoed
f.d.hoogmoed@students.uu.nl

April 2023

1st supervisor: Dr. I.R. Ioana Karnstedt-Hulpus
2nd supervisor: Prof. dr. Yannis Velegrakis
ING supervisor: Frits Hermans MSc

Department of Information and Computing Science

Universiteit Utrecht

Acknowledgements

I am extremely grateful to ING Bank N.V. for the opportunity to research this topic in the context of a real world application in the banking sector, with special thanks to Frits Hermans MSc for proposing the topic and taking on the role of external supervisor. Furthermore, the meetings and frequent days at the office proved invaluable for the experience and insights into the topics and challenges regarding the real world application. Moreover, I would like to thank my supervisor Dr. I.R. Ioana Karnstedt-Hulpus for the detailed feedback and many educational discussions. I would also like to extend my sincere thanks to Prof. dr. Yannis Velegrakis for taking on the role as second supervisor and reader of this thesis.

Special thanks to my girlfriend who took the time and effort to fully read the thesis and provide helpful feedback, but also for her continued support throughout the entire process of this thesis. I am also thankful for my brother who fully read the thesis to provide helpful insights and input. Many thanks to my parents for their support and belief for the entire duration of the thesis.

Abstract

Entity resolution is a critical task in enhancing data quality and ensuring the reliability of data analytics, as it involves identifying distinct records in a dataset that correspond to the same real-world entity. Despite the development of numerous systems to address this challenge, entity resolution remains a complex problem. In this paper we survey and evaluate different approaches for grouping records into real-world entities. Furthermore, we introduce the Predictive Cluster Algorithm Selection (PCAS) method, which selects different clustering approaches for subsets of the data and combines them to produce a unified clustering result. Additionally, we conduct experiments using weighted and unweighted ensemble clustering as alternative approach for integrating diverse clusterings. Our findings indicate that multiple existing clustering algorithms are effictive for the task of entity resolution. Furthermore, we show the potential of PCAS to outperform other approaches and demonstrate the strength of ensemble clustering.

Contents

1 Introduction

Big data is a concept that has become indispensable in the current day and age. This is in part due to the high velocity with which data is created and stored [1]. Furthermore, it has become easier to gather data using the internet as it is ever growing with information and thus a rich source of (un)structured data. However, a problem described long before the inception of the internet has become more prevalent with this growth. This is the problem of *Entity Resolution (ER)*, which is the task of finding records in one or more data sets that describe the same real-world entity. Performing entity resolution is for example useful in cases where biased data has to be detected and prevented when performing analysis on the data [2], but also in cases where data from different sources need to be linked on a common field, which is often done with biomedical data to conduct epidemiological studies [3].

Therefore entity resolution is also known as data matching, record linkage, duplicate detection, deduplication and more names in this general direction. Nonetheless, there are subtle differences that can be found. The name *record linkage* is often associated with the problem of combining two databases or data sets, where a record from one database can either be linked to the other if a match is found indicating they refer to the same entity, or not linked and thus it is an entity not found in both databases. This assumes that there are no duplicate entities in both individual databases, often referred to as *clean* data sources. Therefore, data sources that do contain duplicates are *dirty* data sources, with the major difference being that linking two clean sources guarantees that at most there are two records per entity, and the maximum amount of entities being the sum of both record counts. Whereas with a dirty source (or multiple dirty sources combined) the maximum amount of entities is the same, but the amount of records per entity is unknown. In this paper the problem will be referred to as *Entity Resolution* or *ER*, where the main focus is on dirty data sources.

1.1 Research questions

The growing amount of stored data also requires better methods for entity resolution to keep the data clean. For example, online retailers that expand their product catalog regularly do not want to manually go through their catalog each time they add new products, to prevent having duplicate items in their catalog[1]. To prevent duplicates in data sets or group the duplicates present in a data set, a multitude of entity resolution systems [4, 5, 6, 7] are created. A common approach uses multiple steps to find the duplicates, with the final step being one that groups the duplicates together, also referred to as *clustering*. As there are many steps to such a system, with even more different methods to perform each step, it is difficult to know what system to use and which methods to apply.

[1] `https://www.nchannel.com/blog/challenges-ecommerce-catalog-management/`

Therefore in this work, we aim to explore and evaluate different methods for the clustering step of duplicate items in a data set. We formalize this aim as follows:

RQ1: *How do different clustering algorithms perform on the task of entity resolution and which algorithm is an overall good choice for the task of clustering in an entity resolution system?*

As other papers on entity resolution [8, 9, 10] focus on similar types of clustering algorithms, we make use of multiple different existing algorithms with different types of clusterings to solve entity resolution problems. By also using data sets proposed in different works[11][2] we can evaulate the performance of different algorithms compared to those in other works. Furthermore, this paper looks to well known algorithms lesser used in the context of entity resolution, to get an idea of their performance compared to other algorithms in this context. To do so, an entity resolution system is set up which performs the task of entity resolution using multiple different steps. Such a system is also called an *entity resolution pipeline*.

The second aim of this paper regards using the different types of algorithms together to come to a new combined clustering:

RQ2: *Is it possible to leverage the strengths and different types of clusterings to predict which clustering algorithm to use on a component, based on features of that component?*

Other works that aim to improve the results of the clustering, propose different and new clustering algorithms [12, 13, 10] that aim to algorithmically increase the accuracy of the resulting clusters. The work by Muhammad and Van Laerhoven [14][3] introduces a classifier that predicts which algorithm to use. However, that is in the context of community detection with a limited amount of algorithms to choose from. Therefore, in this paper we will increase the amount of algorithms used for the classifier, as well as applying the classification to the entity resolution problem.

Furthermore, combining results from different clustering algorithms is studied in the field of *ensemble clustering*[15]. As this paper is researching leveraging strengths of different clusterings, the aim is to find how ensemble clustering performs in the context of entity resolution. Furthermore, the possibility of assigning weights to the clustering result of well performing clustering algorithms is explored to further increase the performance of the final clustering. This is summarized in the following research question:

[2] `https://people.cs.umass.edu/~mccallum/data.html`

[3] `https://github.com/SyedAgha/Divide-and-Conquer/blob/master/DCS_code_and_paper/DCS.pdf`

RQ3: *How well does ensemble clustering perform in the context of entity resolution and can the performance of ensemble clustering be improved by assigning weights to individual clustering algorithm results?*

Chen et al. [16] explore the use of ER ensemble, where they combine the results from different entity resolution systems to get to a new result. This means that the final result is solely based on the results of entire systems, meaning that there can be many different preceding steps to get to the final result. Therefore in this paper, we want to focus on ensemble clustering in the final step, to get an understanding how and if the clustering result can be improved with the input that is given to the clustering step.

1.2 Contributions

The main contributions of this paper are:

An exploration and evaluation of clustering algorithms in the context of entity resolution. By using algorithms that are already used in entity resolution, but also taking clustering algorithms from different research area's, we provide an overview of cluster performance of a wide range of different types of clustering algorithms.

Furthermore we propose PCAS: Predictive Cluster Algorithm Selection, which is a classifier that predicts which clustering algorithm to use on different parts of the data set, based on different properties of the data set parts. By doing so, it allows the exploration of different types of clusterings within a single data set, possibly improving on the usage of a single clustering algorithm per data set.

Lastly, we look into ensemble clustering for entity resolution and experiment with a simple method to create a weighted version of the ensemble clustering.

In short, our results show that there are different algorithms that are capable of obtaining a good result on known entity resolution tasks. Moreover, it shows which algorithms are not well suited for the given data sets, which might also provide an insight as to what type of algorithm to use for each task.

The PCAS method shows the potential that it is possible to measure performance gains in terms of accuracy compared to the individual algorithms. However, PCAS is sensitive to the performance of the algorithms available for its selection and thus also sensitive to the parameters used for each parameterized algorithm. This results in a theoretical lower/upper limit for the final clustering on the complete data set: as PCAS does not alter any of the clusterings, it is only as good as the best available clustering per part of the data set if it perfectly predicts which algorithm to use per part of the data set. Furthermore, the selection of features used for the predictive model in PCAS are found to be

important. Without features that distinguish clearly between different parts of the input graph, it is hard for the model to learn which algorithm performs well on which part.

Weighted ensemble clustering shows potential for entity resolution by obtaining results very close to that of the best performing algorithm. However, it lacks consistency in terms of its results, being outscored by the unweighted ensemble clustering on multiple occasions.

The rest of this paper is structured as follows: Section 2 discusses the background and related work on entity resolution, with an emphasis on clustering. Accordingly, section 3 explains PCAS and (un)weighted ensemble clustering. Moreover, section 4 demonstrates the experimental setup with the results presented in section 4.4. Finally, in section, 5 one can find the conclusion and a discussion on future work.

2 Background and Related work

First, this section will discuss background knowledge in relation to entity resolution, including for instance detailed information about different clustering algorithms that will be used in our proposed methods. In addition to this, a discussion on various concepts used for network analysis is provided in relation to clustering algorithms. This section forms the basis of knowledge used to create our entity resolution system. Next, the related work section will contain relevant works in multiple related research areas, as well as other works that evaluate comparable entity resolution problems or use a similar methodology as our proposed methods.

2.1 Background

2.1.1 Blocking

The first step in an entity resolution pipeline is often the blocking step. This is necessary as the naive approach of comparing all records to each other scales quadratic to the size of the input. Blocking is the process of grouping the records together based on a key (called a blocking key[17]) to reduce the number of pairwise comparisons by only doing pairwise comparisons within a block. Therefore, these blocks that are made need to adhere to two important criteria as defined by Christen [18]: firstly, all matching descriptions are placed in at least one common block. The second criteria is that it has to minimize the number of comparisons. Blocking should find a good trade-off between the two criteria: placing all records in one block would adhere to the first criteria, but not to the second. Reversely, placing every record within its own block would greatly reduce the amount of comparisons done (by not doing any at all), but would completely fail to adhere to the first criteria. The importance of good blocking is in the fact that it is at the very start of the pipeline. If the blocking step puts two matching records in different blocks, the pair will probably not be considered in any of the future steps in the pipeline, thus missing out on a match.

One of the approaches for blocking is that of *token blocking* by Papadakis et al.[19]. They propose a method in which a record is broken down into tokens, often times the individual words present in a record. Then a block is made for each token in the entire collection and the records that contain the token are added to the block. Thus every record is added to multiple blocks, which is good for the first criteria mentioned before by Christen [18], as every record will be in one or more blocks with matching descriptions. However, as each record is put in multiple blocks, it increases the amount of comparisons done which contradicts the second criteria. Even more problems arise as different records that have more than one matching token are compared multiple times, but in different blocks. Therefore, the approach also tackles this problem by propagating the comparisons already done. This means that if a comparison is

already done in a block, another block that contains the same two records to be compared, reuses the result from the earlier block.

Token blocking is an overarching approach, where the concept of token blocking is used in different ways to improve or adjust the results of it. *Attribute Clustering Blocking* [20] is one of those approaches that has token blocking at its core. This approach firstly clusters the records based on attributes that have similar values into non-overlapping clusters, which they call attribute clusters. Subsequently, the token blocking technique is used on these clusters. It creates more blocks, but with less records in them, which the authors claim is on average smaller and therefore reduces the amount of comparisons done.

Derived from token blocking is the work of Papadakis et al. [21]. Their work is inspired by token blocking in combination with their observation of many URI's (Unified Resource Identifier) for web data. These URI's often have high schema heterogeneity which is why they proposed *Prefix-Infix(-Suffix) Blocking*. By creating blocks based on the prefix, infix and suffix of the URI's, they found that due to the schema heterogeneity, this blocking technique created blocks that performed well on the two mentioned criteria by Christen [18]. Even though this technique works well for URI's, it is less performant for other types of data.

Nonetheless, this technique is applicable on less structured data too, in the form of *Q-grams Blocking* [18]. Q-grams converts the token blocking keys in sub-sequences of q characters. Then for every unique sub-sequence a block is made in which all records are placed that contain that sub-sequence, which is similar to regular token blocking with the added benefit that records do not have to contain the exact same strings to be put in the same block, as long as a sequence of q characters are the same.

There are many more blocking techniques [17, 18, 20, 22] which all have their use cases based on the desired metrics that they want to achieve.

At this point in the pipeline, there are blocks consisting of records that are more likely to match with other records within their block than with records from different blocks. Whether or not the pairs in a block match is investigated more in depth in the next step of the pipeline.

2.1.2 Pairwise comparison and similarity calculation

The earlier works by Newcombe et al. and Fellegi and Sunter [4, 5] did not use a full entity resolution pipeline yet. They did already use blocking as their first step, as Fellegi and Sunter [5, p. 1196] mention that a full pairwise comparison would outstrip the economic capacity of even the largest and fastest computers (at the time). However, the result of this step was for them final: either two records were in the match-class or they are in the non match-class. They used the *Naive Bayes* approach for this, by calculating the likelihood that a

comparison vector is part of either classes. To know the distribution of both classes, they needed a prelabeled training set of recorded pairs. Then when a new comparison vector was presented, the model trained using the training set would determine in which class the vector belongs.

More approaches would also determine whether or not a pair is a match in this step. The rule-based approach proposed by Wang and Madnick [7] would have the rules serve as a kind of "key" for a record. A rule would then state something about the value of an attribute and assign a key based on that attribute. For example the rule *"If age > 18 Then title = adult, Else title = child"* would group entities using this as key. Then using more keys would further shrink the group till a group would only have those entities that would be a match.

This idea is further expanded by Hernández and Stolfo [23] where the rules would be a bit more complex. Moreover, they use terms like "reasonably close" or "differ slightly" in the context of comparing two fields. That allowed for rules that state: *"If name from record A and the name from B differ slightly, and the address from both records match exactly, they are the same person"*. However, as the rules became more complex, so did the amount of manual labour involved in creating the rules.

Furthermore, by introducing the terms for similar fields, the authors also touch on the subject of string similarity or distance-based techniques. Rule-based techniques are a case of distance-based techniques, where the rules determine the distance to be either 0 or 1. For distance-based techniques this value can often be anything in between 0 and 1, indicating *how* similar two records are. Many different approaches are taken to determine this similarity value for two strings. One well known is the Levenshtein distance [24], which calculates an edit distance needed to transform one string into the other, using insertion, deletion or replacement of characters. Another metric often used is the Jaro-Winkler distance [25], which differs from the Levenshtein distance due to the Jaro-Winkler distance being a measure of the characters that two strings have in common.

More field matching techniques are discussed in the survey by Elmagarmid et al. [26]. The final result of this step can be represented as a graph, where the records are the nodes of the graph, and the edges between the nodes indicate that they had their similarity score calculated, with an edge weight equal to the similarity score. This graph, often called the similarity graph, can then be used for clustering, to find the nodes that are actually the same entity. Often times, edges from this graph are cut based on a threshold set for the similarity score. This threshold can be used to decide that a pair of nodes is the same entity or not, based on the existence of an edge.

2.1.3 Clustering

When considering an entity resolution pipeline with clustering as final step, it is important to know what kind of clustering needs to be done. This is due to the fact that there exists a wealth of algorithms all with different aspects and attributes. In the context of entity resolution, it is important to look at clustering algorithms that for example do not require any prior knowledge, also known as unconstrained algorithms. These algorithms do not have to have the amount of clusters to be found as an input, or other domain specific parameters.

Furthermore, the field of community detection algorithms contains multiple algorithms that do not require the amount of clusters as input, which creates a clustering based on the network of communities [27]. For this paper, we will use the term *graph* instead of *network* and *clusters* instead of *communities*, as these terms are interchangeable when it comes to the concept that they describe. The use of community detection algorithms in the context of entity resolution is not new, as McConville et al. [28] use an ensemble of community detection algorithms for deduplication of vertices.

For our approach, we cluster on the input graph given by the step described in section 2.1.2. The blocking step and the application of a score threshold after the similarity calculation may result in the graph being split into multiple smaller subgraphs. To detect these subgraphs, an algorithm called Connected Components (also known as Transitive Closure or Partitioning) [29] can be used. Connected Components works by initially making each node V its own cluster. Next a scan is done over the edges in the similarity graph, merging the clusters of two nodes if they have an edge between them in the similarity graph. Intuitively, this results in large clusters containing lots of nodes, improving the recall but lowering the precision. In early work of entity resolution, this was the most common approach for creating clusters, as it can be done efficiently in a single pass over the nodes in the graph G. Furthermore, it provides a feasible result, albeit on the less accurate side. Another observation about Connected Components is the fact that the weight of the edges is not taken into account, only whether an edge exists or not. It is still often used as a baseline to test entity resolution results against, but more often it is used as the basis for other algorithms. Connected Components is performed first to then perform a different clustering algorithm on the given component, or it is used with a small alteration to the cluster generation.

As we specifically want different clusterings to use for PCAS, we use community detection and clustering algorithms based on the different approaches that they take to cluster the nodes in a graph. We roughly group the approaches in the following categories; Hierarchical clustering methods, Modularity optimization clustering methods, Random walk-based clustering methods, Structural similarity-based clustering methods.

Hierarchical clustering methods

Hierarchical clustering (abbriviated to HCL) is an old clustering algorithm, with the method proposed by Sibson [30], forming the basis for many other hierarchical clustering algorithms. This group of algorithms can be split in to agglomerative and divisive clustering algorithms. For agglomerative algorithms, the general idea is to make each node its own cluster. Then based on some distance function, it decides whether two nodes are part of the same cluster. For example, well known is the *minimum distance* also called *minimum or single linkage* distance function. The algorithm takes a node and calculates the distance to all other nodes, pairing the selected node with another node that has the minimum distance to it. This continues for all clusters, where clusters with multiple nodes use the minimum distance to any of its nodes as distance measure. The result is a hierarchy of clusters, where the smallest clusters at the bottom are the closest together, while the root cluster contains all nodes in it. The same process can be done in reverse for divisive clustering, starting in a large group and splitting into two clusters that are the furthest apart. The algorithm itself has not changed much, but the distance function used has. In the analysis by Jarman [31] for example, they compare four different types of linkage methods: the earlier mentioned single linkage, complete linkage, average linkage and centroid linkage. Each name already already indicates to a certain extend how the linking works, which mainly takes either different nodes within the clusters (for single, complete and centroid) or the average of all nodes within the clusters.

One of the main issues with HCL is the fact that for the linkage methods it is required to calculate this distance between all nodes. Therefore, without the usage of an earlier step like blocking, it would not be feasible to perform HCL on large data sets. But even with blocking this problem could occur when a block is too large. Hierarchical clustering is however still often used and improved to be able to work for larger scale entity resolution [12, 32].

Affinity propagation has not often been associated with entity resolution, but has been used more in different tasks related to entity resolution [33, 34, 35]. It was originally proposed by Frey and Dueck [36] as another unsupervised clustering algorithm that did not need to know the amount of clusters that it has to create. Instead, it works by sending messages between nodes, informing those nodes about the relative 'attractiveness', which in the context of entity resolution has to do with the similarity score. Then each receiving node responds to the sending nodes with its willingness to be associated to the sender, based

on the received scores. This prompts those first sending nodes to update their relative scores after which an iteration is complete and a new one can start. Once a consensus is reached after multiple iterations, the actual clustering is made based on the nodes that each node associates to. If multiple nodes associate with the same node, they are placed within the same cluster. By using this iterative approach of clustering nodes together at different stages, we assign it to the hierarchical group.

The OPTICS (Ordering Points To Identify the Clustering Structure) algorithm is an algorithm proposed by Ankerst et al. [37]. It is a density-based clustering method that is suited for discovering clusters with varying densities and shapes, where the density is defined as the distance between nodes, hence the grouping with other hierarchical approaches. The intuition behind OPTICS is to model the density of data points in a way that enables the detection of clusters without prior knowledge of the number of clusters.

To achieve this, it uses two main steps in the algorithm. Firstly, it creates a reachability plot by computing the reachability distance between points in the dataset. This reachability is the minimum distance required for a point to be directly density-reachable from another point. The distance is based on two configurable parameters, the neighborhood radius and the minimum number of points required to form a dense region. Points are then ordered based on their distance, resulting in an ordering that reflects the underlying cluster structure. The second step is to extract the clusters from the reachability plot, by identifying valleys, which represent transitions between dense regions. OPTICS can be an effective clustering algorithm in the context of entity resolution due to the ability of finding clusters of varying densities and shapes, as the similarity graph could contain clusters of many different types.

The DCS (Divide and Conquer Strategy) by Muhammad and Van Laerhoven [14] is an algorithm that aims to divide the input graph into smaller subgraphs and cluster by selecting a leader based on features of the subgraph. Then the clusters around the leaders are expanded using a scoring function, iteratively adding or removing nodes. Lastly, the clusters obtained on each subgraph are merged to get an overview of the clusters in the input graph. They argue that 'it is a widely accepted observation that complex systems exhibit hierarchical organizations, in which each module represents a bigger picture and it further contains a set of nested communities' [14, p. 2]. This is why being able to properly detect the subgraphs and perform clustering on those subgraphs, give a good indication of the total final clustering.

Modularity optimization methods

The Louvain algorithm [38] is a popular and efficient community detection algorithm that seeks to optimize the modularity. Modularity is a measure proposed by Newman and Girvan [39] that quantifies the quality of a network partition. This is defined as the difference between the observed fraction of edges within

communities, and the expected fraction of edges when edges were to be placed randomly while preserving the degree of each node. The Louvain algorithm operates through a hierarchical agglomerative approach that consists of two main phases which are repeated until the modularity converges. The first phase is the local optimization phase, where each node in a community is greedily moved to the neighboring community that yields the highest gain in modularity. In the second phase, the communities are aggregated where each community is collapsed into a single node, forming a new graph. The Louvain algorithm is found more often in the context of entity resolution [40, 12, 28] due to how popular it is in the community detection research area. Furthermore, the algorithm is fast and scalable to large graphs, but might be prone to getting trapped in a local optima due to its local optimization phase.

The algorithm proposed by Sobolevsky et al. [41] also makes use of optimizing the modularity measure for higher quality community detection. Sobolevsky et al. start by selecting an initial partition made of a single community. After that, the following two steps are repeated as long as any gains in terms of the modularity are observed. The first step is to find for each source community (the communities at the start of this step) the best possible redistribution for every source node (the nodes in the community at the start of this step) into destination communities, which is calculated using the modularity. These destination communities can either be an existing community or a complete new one. Then in the second step, the best merger/split/recombination of communities is performed again based on the modularity. Due to using a combination of different steps to get to its final clustering, the algorithm is called *Combo*. The characteristics of the algorithm align with the goals of entity resolution, by optimizing the modularity it seeks to identify groups of entities that are densely connected within the cluster while being sparsely connected outside the cluster. Compared to the Louvain algorithm, Combo differs in its approach to optimize the modularity by not aggregating the communities after each iteration. Furthermore, due to this difference in optimizing the modularity, Combo is less likely to get stuck in a local optimum. This however comes at the expense of using an iterative optimization approach which leads to high computational complexity when using larger graphs.

Clauset et al. [42] proposed a fast and efficient hierarchical agglomerative clustering algorithm for detecting community structure in large networks, and could thus be described as either a hierarchical algorithm but also as a modularity optimization algorithm. The Greedy modularity algorithm is, similar to the approaches by Blondel et al. [38] and Sobolevsky et al. [41], based on the concept of modularity optimization. The key intuition behind the algorithm is that it seeks to merge pairs of communities that result in the largest increase in modularity, and this process is repeated until no further improvement is possible. The main difference is found in the way the modularity is optimized. Blondel et al. use a local optimization by only looking at neighbor-

ing communities per node, followed by community aggregation for optimizing the modularity. The method by Sobolevsky et al. uses a combination of merge, split and recombination of communities to maximize the modularity, whereas Clauset et al. adopt a greedy search approach to iteratively merge pairs of communities in a hierarchical manner. As Greedy modularity shares the same modularity based objective as the other modularity optimization methods, its results are also aimed at identifying the densely connected clusters. Although the aim and intuition of the algorithms are the same, the multiple different modularity optimization approaches may result in different types of clusterings.

Random walk-based methods

The theorem for Markov Clustering (abbrivated to MCL) by Van Dongen [43] is based on random walks on a graph. When random walking on a graph and keeping track of the nodes visited, the intuition is that the random walk will visit clusters (and especially cluster centers) a lot more than nodes that are further away or less connected nodes. Simulating the random walk by actually performing the random walk on the graph would be too expensive. Instead, Van Dongen [43] managed to simulate it by performing simple algebraic operations on the matrix of edges. Firstly a normal matrix product or otherwise called *expansion* is applied to the matrix. This step models how the walk would spread around the graph. Then an inflation step is applied that calculates the Hadamard power, which models how the flow gets stronger in densely connected parts and weaker in sparser parts. The algorithm performs these steps a number of iterations to converge to a clustering. In the framework by Hassanzadeh et al. [9] a number of algorithms are evaluated to perform entity resolution and see how every algorithm performs. Their evaluation and final conclusion shows that MCL is highly scalable, has a good ability to find a correct clustering and is capable of dealing with different types of data sets [9, p. 1292].

Walktrap is introduced by Pons and Latapy [44] as a community detection algorithm based on the principle of random walks in a network. The idea is that a random walk in a network tends to remain within the densely connected communities. By performing this random walk, the algorithm computes a distance metric between nodes that reflects the likelihood of a walk transitioning between them. This distance metric is then used to hierarchically cluster nodes using a different hierarchical clustering algorithm. The intuition of using a random walk has been proposed before, for example the Markov clustering method [43] as explained in the previous paragraph. However, Pons and Latapy [44] perform random walks of a set length and hierarchically cluster the result, whereas Van Dongen [43] simulates the random walks using iterative matrix operations and pruning of the result to get to a clustering. Because the Walktrap algorithm uses a set length for the random walk, it is also sensitive to this parameter in terms of the accuracy of the detected clusters. Furthermore, using a hierarchical clustering algorithm to obtain the clustering can be a potential bottleneck for large networks, although it also provides granularity in the results.

The Diffusion Entropy Reducer (DER) proposed by Kozdoba and Mannor [45] is also a community detection algorithm based on random walks. They observe that nodes belonging to the same community exhibit similar connectivity patterns, and thus their probability measures, derived from random walks, can be viewed as points in a measure space. By embedding these points into a lower-dimensional Euclidean space, the algorithm captures the underlying community structure by clustering nodes with similar embeddings. This is also the difference with Walktrap, as that computes a distance metric based on the proximity found by the random walks after which the nodes are hierachically clusterd based on the computed distances. In contrast, DER embeds the points in a measure space and applies the k-means clustering algorithm. To the best of our knowledge, there is limited related work on the direct application of the DER algorithm in the context of entity resolution. However, given its ability to capture the connectivity patterns in networks through random walk-based embeddings, it has the potential to perform well when applied to entity resolution tasks where the relationships between entities can be represented as complex networks.

Structural similarity-based methods

SCAN is a Structural Clustering Algorithm for Networks, prosposed by Xu et al. [46] that focuses on identifying communities based on the structural similarity rather than the density of the network. They argue that nodes within the same community exhibit similar structural patterns. Xu et al. define structural similarity between two nodes based on the concept of neighborhood overlap. The neighborhood overlap between two nodes measures the similarity of their local neighborhoods. Specifically, the structural similarity is defined as the ratio of the number of common neighbors they share to the number of nodes in their combined neighborhood. That is also what differentiates SCAN from other algorithms, as it does not have a random walk or a modularity based approach to finding the clusters, unlike the previously mentioned algorithms in this section. The full algorithm works by calculating the structural similarity between all nodes and its neighbors. Next, using a predefined threshold ϵ, neighborhoods are created where all nodes have a structural similarity higher than ϵ. Finally, a density-based clustering algorithm is used on all the neighborhoods to identify the clusters, which requires another parameter that sets the minimum amount of required nodes to form a cluster. Using this distinct approach might result in different clusterings compared to other algorithms, as there might be entity resolution tasks where the structure of the graph plays an important role in finding the correct clustering. However, the performance of SCAN may be sensitive to the choice of ϵ and the minimum number of nodes required for a cluster.

Li et al. [47] proposed Identifying Protein Complex Algorithm (IPCA) in the context of finding protein complexes in protein-protein interaction networks. The key idea is to identify clusters based on maintaining the diameter of a

cluster, which is the maximum shortest distance between all pairs of vertices. This is done by firstly weighing the edges based on the shared neighbors of nodes. Then dense subgraphs are identified (called *cores* by Li et al.) based on iterative selection of edges with the highest weight, and merging adjacent cores. Lastly, the cores are expanded by adding the remaining nodes that have a strong connection to a core. The resulting clustering emphasizes the importance of strong connections in dense connected parts of a subgraph. Therefore, in the context of entity resolution, it could help in finding tightly knit clusters when a large connected component is presented. One of the drawbacks, however, is the fact that IPCA creates overlapping clusters, which goes against the intuition that a cluster in entity resolution stands for a single entity.

There are many more clustering algorithms already used in the context of entity resolution [17, 9] and even more algorithms on clustering in general. Nonetheless, for the purpose of this research, the algorithms discussed in section 2.1.3 are evaluated to compare known entity resolution clustering algorithms to other clustering and community detection algorithms.

Ensemble clustering

Ensemble clustering is a technique used to enhance the robustness and quality of clustering results by integrating multiple clustering outcomes without accessing the features or algorithms. Ensemble clustering can improve clustering quality, achieve robustness, novelty, stability, and support distributed computing and scalability. The approach has two stages; Diversity, where base clusterings are generated, and a Consensus function, which combines the base clusterings to obtain the final solution. The quality and diversity of base clusterings significantly impact the performance of the clustering ensemble. However, achieving the right balance of diversity and quality requires post-analysis. The combination of multiple clusterings using a consensus function is a challenging task, as there is no predefined labeling, and the labelings in base clusterings are virtual or not real. The survey by Golalipour et al. [48] provides a comprehensive overview of many different ensemble clustering techniques.

2.1.4 Features

The paper by Muhammad and Van Laerhoven [14] mentions the use of properties that are used to describe network topology. As networks are represented as graphs, we consider the same features used. For this section, we assume an undirected, weighted graph $G(V, E)$ where V are the *vertices* or otherwise called *nodes*, and E are the edges. An edge $e = (u, v)$ consists of two nodes such that $u, v \in V$. Furthermore, weights are defined as a function $W(e)$ where $e \in E$. The *degree* of a node v is defined as the amount of other nodes that it is connected to. Another important definition, is that of *triangles* in a graph. Whenever three nodes are connected by edges, for example nodes u, v, w by edges $(u, v), (u, w), (v, w)$, it is called a triangle. Triangles are found on a per

node basis, which means that node u finds the triangle of nodes u, v, w, but also node v finds the triangle v, w, u. Therefore, when counting the triangles over the entire graph on a per node basis, each triangle is counted 3 seperate times.

The *average clustering coefficient* for weighted graphs is a property of a graph that describes the extent to which nodes in a graph tend to cluster together. It is defined as the geometric average of the subgraph edge weights [49] and calculated per node using equation 1

$$c_u = \frac{1}{deg(u)(deg(u)-1)} \sum_{vw} (\hat{w}_{uv}\hat{w}_{uw}\hat{w}_{vw})^{1/3} \tag{1}$$

where edge weights $\hat{w}_{uv}$ are normalized by the maximum weight in the network using $\hat{w}_{uv}/max(w)$, and edges $(u, v), (u, w), (v, w)$ form a triangle in the graph. Furthermore, $deg(u)$ is the degree for node u.

Transitivity is a measure similar in intuition to the clustering coefficient, describing the tendency of nodes to cluster together. It is based on the relative number of triangles in the graph, compared to the total number of possible triangles in a graph. The transitivity T is calculated using equation 2.

$$T = \frac{3 * \#triangles}{\#possible\_triangles} \tag{2}$$

where #triangles is the number of triangles found in the graph and #possible_triangle the amount of possible triangles. The factor of three accounts for the fact that each triangle contributes to three different possible triangles [50].

Density d measures how densely connected a graph is based on the amount of edges compared to the amount of possible edges. The amount of possible edges is defined as the amount of edges the graph would have had if every node had an edge to every other node. For undirected graphs, this is calculated using equation 3.

$$d = \frac{2m}{n(n-1)} \tag{3}$$

where n is the number of nodes and m is the number of edges in G.

The *diameter* and *radius* are properties of a graph that describe the spread of a graph. With the diameter being the "longest shortest path" within the graph. This means that for every node u the shortest path is found to every other node. Next, the largest value in these shortest paths, is the maximum distance from node u to any other node, with the diameter being the longest maximum distance for the whole graph. Similarly, the radius is the lowest value out of all the maximum distances for the nodes in the graph. The node(s) that have this lowest value, are often nodes found at the centre of the graph

To find the nodes at the centre of a graph, the *harmonic centrality* by Boldi and Vigna [51] can be used. They make use of the distances calculated for the shortest paths, which can be seen in equation 4.

$$C(u) = \sum_{v \neq u} \frac{1}{d(v,u)} \tag{4}$$

where $d(v,u)$ is the shortest-path distance between nodes v and u. As this is calculated on a per node basis, the average of all the harmonic centralities can be used as a feature for a graph.

2.1.5 Machine learning models

This work and previous works on entity resolution often make use of one or more machine learning models. Machine learning (ML) is a subset of artificial intelligence (AI) that involves the development of algorithms and computational models that enable computers to learn and adapt from experience, without being explicitly programmed [52]. The primary aim of ML is to allow computers to generalize from limited data, recognize patterns, and make predictions or decisions autonomously [53]. Linear regression, logistic regression, and active learning are essential machine learning techniques used for various predictive modeling tasks. In this section, we will briefly describe each method and their applications.

Linear regression is a supervised learning algorithm used for predicting a continuous target variable based on one or more input features [54]. It models the relationship between the input features and the target variable using a linear function. The algorithm's primary goal is to minimize the sum of squared differences between the predicted values and the actual values, also known as the least squares criterion [55]. Linear regression is widely used in various fields, including economics, finance, and social sciences, for forecasting and understanding relationships between variables.

Logistic regression is a supervised learning algorithm but is often used for binary classification tasks, where the goal is to predict one of two possible classes for a given input [54]. Instead of modeling a linear relationship between input features and the target variable, logistic regression models the probability of an input belonging to a particular class using the logistic function [56]. Logistic regression can be extended to perform multi-class classification, which is called multinomial logistic regression. Similar to logistic regression, multinomial logistic regression models the probability of an input belonging to each class. However, instead of using the logistic function, it employs the softmax function to estimate class probabilities [57].

Active learning is a semi-supervised learning paradigm that aims to reduce the amount of labeled data required for training machine learning models [58].

In active learning, the algorithm actively selects the most informative samples from a pool of unlabeled data and queries the oracle (e.g., a human expert) for their labels. This process is iteratively performed, allowing the model to learn from the most informative samples and thereby improving its performance with fewer labeled examples [59].

2.2 Related work

Entity Resolution is not to be confused with Entity Recognition or Entity Disambiguation. Entity recognition mainly focuses on identifying named entities in text [60], whereas entity resolution is more focused on identifying multiple mentions of entities as the same entity. Moreover, the goal of both research fields are different: entity recognition identifies types of entities for the use case of categorizing text or for extracting content automatically whereas entity resolution looks for a result set of matched entities from a large set of (possible) duplicate entities. Then there is the domain of entity disambiguation (or entity linking) which aims to link an unique identity to an entity in text mentions. This makes use of a knowledge base containing the unique identities that the entities should be linked to. This is particularly useful for cases where the same word can have different identities. An example is the word Apple. This could be the fruit, the company or less likely but still possible, part of the nickname for New York: The Big Apple. Again, the difference is in the goal of the research field and is therefore not to be confused with entity resolution.

Entity resolution solutions have different approaches to reach their end goal, mainly based on the different types of problems tackled. Earlier works from Newcombe et al. [4] and Fellegi and Sunter [5] were performed on record linkage. They modeled the problem of record linkage as a statistical problem where a comparison vector (comparison between two records) is the input of a decision rule, that either assigns the vector to the "match" class or "no match" class based on the probability of assigning the vector to either of the classes. Modelling the problem in this manner created the basis for the *Probabilistic matching models*.

Some other techniques proposed use the generation of multiple partially identifying keys for a record, to then be able to match other records based on similar keys [6]. Others use rule-based approaches where rules have to be defined by experts that can tell instances apart based on those rules [7]. However, this required many rules to get to a concise entity resolution result. Therefore, Sarawagi and Bhamidipaty [61] designed a learning-based deduplication system that uses a method of interactively discovering challenging training pairs using active learning. These training pairs could then be used to train a classification model that classifies record pairs as duplicates or not.

In the end, all techniques boil down to the same template of steps that have to be taken towards the end goal, often referred to as an entity resolution

pipeline. In the first place it is decided which records in the data source are candidates that might be matched against each other, also known as blocking. Then these candidate pairs are evaluated using some measure to understand how related they are. After that, a decision has to be made whether these candidate pairs are actual matches or not. However, more steps can be added to improve for example the efficiency or accuracy. All of the steps can be executed in different ways resulting in distinct outcomes.

Entity resolution has evolved a lot over the years, as each step was researched more in depth. Recently, an extensive overview was provided by Chrisophides et al. [17] where many different methods for each step are briefly touched upon with their respective pros, cons and use cases. However, the evaluation of methods in entity resolution, and more specifically the clustering part, is rare to find. Hassanzadeh et al. [9] compare many different clustering methods, two of which are Connected components as baseline, as well as Markov clustering, on different data sets and different alterations of those data sets. They use well known clustering evaluation measures to compare the result of their entity resolution. Furthermore, to test an entity resolution solution it is necessary to know the perfect existing clusters as otherwise there is no ground truth to test the provided solution against. For smaller data sets it is possible to label the ground truth clusters manually. Nonetheless, this becomes undoable for larger data sets.

Christen [62] thus presented a data set generator that uses a duplicate-free data set as input, inserting erroneous duplicate records to create a data set with known groundtruths. Furthermore, it allows for creation of data sets with configurable amounts and types of errors, but also for the amount of duplicates per record. That is why often papers use the same set of available data sets that are already labeled, either through manual labeling or automatic duplicate creation. Therefore, not just an evaluation of multiple clustering methods was done by Hassanzadeh et al. [9], but also a framework was setup where they publicized the data sets that they used[4] as the benchmark for entity resolution systems. This is necessary as Köpcke et al. [63] noticed that there were some data sets that are often used in entity resolution, yet they are not always consistent when it comes to the size, possibly due to using a duplicate generator with different settings. Having a framework for entity resolution evaluation could help in more consistent evaluation of entity resolution across different researches. In their paper they propose an evaluation framework that consists of four data sets that are prelabeled[5]. This is however for the task of Entity matching, where two data sets are matched against eachother, thus only containing clusters of at maximum two records. Therefore, Saeedi et al. [11] do the same but for multi-source entity resolution. This means that there is still a known number of

[4] http://dblab.cs.toronto.edu/project/stringer/clustering/

[5] https://dbs.uni-leipzig.de/research/projects/object_matching/benchmark_datasets_for_entity_resolution

sources used, but this information can be discarded by an algorithm making it a dirty entity resolution problem. The data sets created by the authors are made by taking a real world duplicate free data set and using the DAPO data generator [64] that creates duplicates with subtle errors. By sharing the data sets[5] it has become the more consistently used benchmark for entity resolution, as the data sets are available on a website[6] where people can 'compete' for better entity resolution results using three of these data sets.

However, as noted by Menestrina et al. [8], oftentimes entity resolution evaluation measures are chosen ad-hoc and are even conflicting with each other. For this reason, the authors research the existing measures and propose their own new measure which is specifically aimed at entity resolution, which they call the generalized merge distance. This measure is an indication of how many splits and merges of clusters are necessary to transform the entity resolution solution to the actual ground truth solution. Other measures that they evaluate are mainly based around the F1 score consisting of precision and recall. However, calculating this precision and recall for clusters can be done in different ways. Menestrina et al. [8] evaluate different F1 score measures; one that is based around pairwise precision and recall, another F1 score that is based on exact cluster precision and recall and finally an F1 score based on cluster similarity precision and recall.

Lastly, they also evaluate a measure that indicates the information lost and gained when the result is compared to the ground truth. They conclude that there are considerable differences between the measures, and that their own proposed method correlates the most to the pairwise F1 score and the information lost and gained measure. However, that does not mean that only these measures should be used, as each measure is based on a characteristic of the resulting clusters. Therefore each measure is still useful depending on the desired metric that needs to be evaluated and advocates for careful selection of which evaluation metrics are used.

Saeedi et al. [11] evaluate some of the same algorithms as in the framework by Hassanzadeh et al. [9]. The difference is that this time parallel implementations are used to measure the scalability in terms of the size of the data set, but also in terms of how parallelizable the algorithms are. They do not explicitly mention using the measure as described by Menestrina et al. [8] but do explain that the metrics that they use are based on the pairwise precision, recall and F1 score. This information is thus crucial when it comes to interpreting these scores.

The paper by Draisbach et al. [10] proposes new algorithms for clustering in the context of entity resolution. The data sets used to test the proposed clustering algorithms are those from earlier works [9, 11], thus indicating more standardization for better comparisons. But also the fact that they selected

[6] `https://paperswithcode.com/task/entity-resolution`

metrics from Menestrina et al. [8], creating a thorough evaluation of their own algorithms in comparison to other popular clustering algorithms.

The work by Muhammad and Van Laerhoven [14] proposes a novel community detection algorithm, but also experiment with algorithm selection in the context of finding community structures. They note: 'it is neither clear how to choose among existing approaches nor is it clear when a particular algorithm performs better than others'[14, p. 9]. The selection of algorithms is not done per data set, instead they split their graph in subgraphs (which they call modules), to then select an algorithm per module. They studied various properties of the modules that are used to describe the network topology, and selected two features that described the topology the best. Next, they train a Random Forests Classifier on self generated benchmark data, where the features are used as the actual data and the algorithm that clusters the data the best as label.

Another way to incorporate multiple results into a final result is by using an ensemble of clusterings [65, 15]. This is often done by using a *corrospondence matrix* [66] between the cluster labels produced by different clusterings. Chen et al. [16] use the notion of ensemble clustering by combining results from different entity resolution systems. Furthermore, they incorporate context features derived from each entity resolution system. This is done due to the fact that they want their entity resolution ensemble solution to have no prior knowledge of the different systems presented to the ensemble. Using the features from the context, they present a method to weigh each solution before the final ensemble.

In summary, the related works discussed in this section provide valuable insights into entity resolution and the various methods that have been proposed to address it. While these works have made significant advancements in the field, they either constrain the types of clustering algorithms used to algorithms known in entity resolution, or propose new ones for specific problems within entity resolution. In this paper we aim to broaden the horizon in terms of the clustering algorithms used for entity resolution, by implementing and evaluating clustering algorithms not often found in the context of entity resolution. Furthermore, by performing a similar approach to selecting algorithms as Muhammad and Van Laorhoven, we aim to increase the accuracy of the entity resolution system. Finally, the paper by Chen et al. propose an ensemble system consisting of multiple entity resolution systems, would require access to multiple entity resolution systems. By performing ensemble clustering using multiple clustering algorithms, we simplify this approach by condensing it to the clustering step. In the following sections, we will delve into the details of our proposed methods, present and discuss the experimental results.

3 Methods

This section starts with an overview of the entity resolution system created for evaluating the different clustering algorithms mentioned in section (2.1). Furthermore, a description is provided on how we perform our PCAS method using the proposed entity resolution system. Lastly an insight is given in how (weighted) ensemble clustering is implemented and simultaneously performed for a combined evaluation of all algorithms, PCAS and ensemble clustering.

3.1 Entity resolution system overview

For the entity resolution system, this paper makes use of the Deduplipy python package[7], which is an implementation of an entity resolution pipeline. This package is chosen as it incorporates the basic steps of a pipeline, consisting of blocking, similarity scoring and clustering. Furthermore, it is highly customizable making it possible to implement more clustering algorithms, as well as implementing PCAS and (weighted) ensemble clustering. Not just the steps are customizable, but also the usage of different functionality within the steps. Builtin are multiple blocking key possibilities, different string similarity functions and the use of an active learning model to predict whether two records are a match based on the calculated similarity. Apart from the mentioned builtin functions, it is also possible to incorporate custom blocking keys or similarity functions, making it possible to recreate entity resolution pipelines from different works.

The final entity resolution system used in this research makes use of all the previously mentioned steps, as well as adding steps into it for PCAS (section 3.2) and ensemble clustering (section 3.3). This starts with training the active learning model that is later used for calculating a probability that two string are a match. The active learning step in Deduplipy works as follows: It first samples pairs from the given data set. Then it calculates the similarity for the sampled pairs, using the given similarity functions. After that, it begins the active learning process by giving itself a couple of synthetic perfect matches. Doing so, allows the model to learn how cases look where the similarity score is very high. Next the model prompts the user with pairs, asking whether the pair is a match or not. The purpose of this is for the model to learn what the similarity might look like for pairs that are and are not a match.

After the active learning model is trained, the blocking step is applied to the data set. The rules used for blocking can be given by the user. If not, Deduplipy has builtin functionality to select the best blocking rules from all of the rules implemented in Deduplipy. To find blocking rules, Deduplipy defines the best rules by using the sampled pairs from the active learning set, and uses the model from active learning to predict which pairs are a match or not. Then

[7] `https://www.deduplipy.com/`

it looks for the rules that groups the records such that the matching records are in the same block. Next, the given or found blocking rules are applied to the data set. This splits the data set into multiple smaller blocks, to decrease the amount of pairwise comparisons necessary in the next step. In the next step, for each block the pairwise similarities for all possible pairs inside each block are calculated, and the active learning model is used to give a probability that each pair is a match. The result of this step can then be filtered by applying a user defined score threshold, which we call Sct, that removes all pairs that have a score below Sct.

With the match probabilities calculated within each block, it is possible to create a graph out of it, using the individual records in the blocks as nodes and the calculated probabilities as weighted edges. The graph created is not a multitude of graphs, but a single graph containing all the nodes in the data set. This is done due to the fact that a node might fall into multiple blocks. The total graph is used as the input graph for the clustering step. At this point we perform the Connected Component algorithm on the graph to obtain all the components present in the input graph. Before we use the individual clustering algorithms on each component, we collect the features of each connected component, in preparation for PCAS.

Then all the clustering algorithms are deployed on each connected component. In our case, this consists of the following algorithms: Hierarchical clustering, Markov clustering, OPTICS, IPCA, DCS, Combo, Louvain, Walktrap, Greedy modularity, DER, SCAN and affinity propagation, as well as keeping the component as its own clustering result. Figure 1 shows a schematic overview of the entity resolution system.

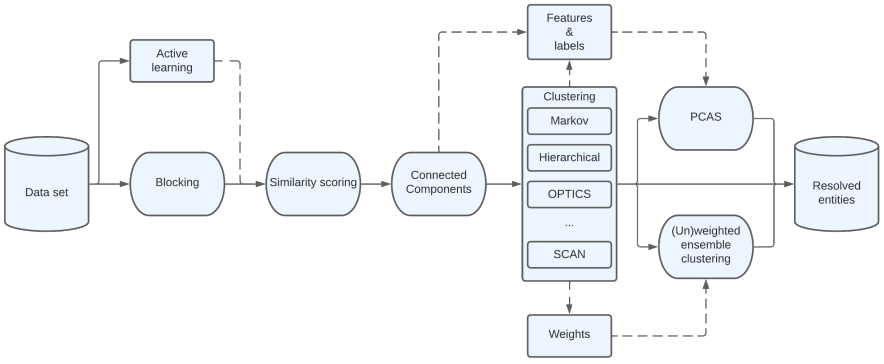

Figure 1: Overview of the entity resolution system. The uninterrupted lines show the flow of the input data, the dotted lines are other calculations used in the system.

The implementations for IPCA, DCS, Combo, Louvain, Walktrap, Greedy modularity, DER and SCAN are provided by Rossetti et al. [67][8]. The hierarchical clustering algorithm is provided by the Scipy python package [68][9]. The python implementation for Markov clustering is taken from Github[10]. Lastly, OPTICS and affinity propagation are used from the sklearn package [69][11]. Together with the clustering step, the ensemble step is also executed, which we explain more about in section 3.3. After this, we perform our PCAS step as described in section 3.2.

3.2 PCAS: Predictive Cluster Algorithm Selection

There is not a "one size fits all" solution to entity resolution or clustering in general. The choice of algorithms used is always highly dependent on the type of data or the desired type of clustering. Therefore, the idea behind PCAS is that by incorporating multiple different algorithms, we remove the need to choose a single algorithm beforehand. Furthermore, by using different algorithms on various parts of the input graph, we investigate whether the resulting clustering accuracy improves by using PCAS. As we do not know which algorithm performs best with which graph features, we train a classifier on the features of components.

The following features are collected and used: average clustering coefficient, transitivity, diameter, radius, nodecount, edgecount, min-max-average edgeweight, density, min-max-average degree, triangles, average centrality. These features are explained in more detail in section 2.1 and are mostly obtained by network analysis or general graph features.

By picking the best performing clustering algorithm for each connected component, we can assign a label to that component's features. The notion of 'best performing' algorithm is used in the sense of its F1 score. Whenever the highest F1 score is equal for multiple algorithms, we look at Generalized Merge Distance as alternative. If the same still applies, we cascade through the metrics until a winner is found. The order in which the metrics are compared, is of great importance, as it decides the winner and thus the label for the training data.

With the features as input data, and the best clustering algorithm per connected component as label for the features, we can train our classifier model. This is a logistic regression classifier as that is a relatively easy to interpret model with fast training time. Using the obtained classifier we can predict which algorithm to use when a new component is presented.

[8] `https://cdlib.readthedocs.io/en/latest/overview.html`

[9] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.fcluster.html#scipy.cluster.hierarchy.fcluster`

[10] `https://github.com/guyallard/markov_clustering`

[11] `https://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster`

3.3 Ensemble clustering

Another method to use the result of multiple clustering algorithms is that of ensemble clustering. This method often tries to find a consensus between multiple clustering algorithms whether two nodes should be in the same cluster. For this paper, we use the *Graph-based consensus clustering* by Yu et al. [70]. The proposed method by Yu et al. creates a so called *co-association* matrix, CO. This matrix is created by creating a similarity matrix for all nodes v in the graph (which in our case is a component). Then each entry in the similarity matrix v_i, v_j represents the edge between node i and node j. Next each clustering algorithm adds a 'vote' to the similarity matrix if they have edge v_i, v_j in one of their clusters. Finally the matrix is divided by the amount of clustering algorithms that are used in the ensemble, to create a normalized similarity matrix. Yu et al. formalize this matrix as follows:

$$CO(v_i, v_j) = \frac{1}{M} \sum_{g=1}^{M} S_g(v_i, v_j) \tag{5}$$

where M is the amount of algorithms and S_g is the similarity matrix by algorithm g. As the resulting matrix is a similarity matrix, we can interpret this as a graph of its own. To get the final clustering, we follow the approach by Yu et al. that removes edges below a given threshold. This threshold is called the *normalized cut* and is applied to the graph obtained from the similarity matrix. In our implementation, we choose different values for the normalized cut to get an idea of the range of solutions possible using ensemble clustering.

Weighted ensemble clustering

The ensemble clustering described in the previous paragraph gives all the different clustering algorithms that participate in the ensemble an equal vote when creating the similarity matrix. However, certain algorithms might perform better for different tasks, which is why we also implement a weighted version of ensemble clustering. In a weighted ensemble clustering, we assign a weight per clustering algorithm that determines how much the clustering result of that algorithm influences the ensemble clustering. The weights can be determined in many different ways [48], however, we chose to experiment with using a linear regression as explained in section 2.1, to learn the weights. For training the linear regression, we loop over the components in the total input graph and look at the different clustering results obtained from the different algorithms. Then we look at all the possible edges in a component and per edge assign a 1 to that algorithm for that edge if the algorithm has that edge in its clusters, and a 0 if the algorithm does not have the edge in its clusters. The groundtruth is used as the actual label of the edge. This is shown in table 1.

When training, we balance the input such that 50% of the training data has groundtruth label 1 and the other 50% label 0. By doing this we prevent a skewed result towards clustering algorithms that put all the nodes into a single cluster or towards clustering algorithms that may separate the nodes into many

Edge	HCL	MCL	IPCA	label
$a \rightarrow b$	1	1	1	0
$b \rightarrow c$	1	1	0	1
$c \rightarrow a$	1	0	1	1
$c \rightarrow d$	0	0	1	0

Table 1: Example of data used as training for the Weighted ensemble clustering linear regression model

smaller clusters. Furthermore, a total of 200 edges are chosen to train on from the training set, as 100 edges per class are assumed to be a sufficient amount for learning. Once the weights are learned using the linear regression, the ensemble clustering uses the same proposed method by Yu et al. The difference is in the creation of the normalized similarity matrix:

$$CO(v_i, v_j) = (\sum_{g=1}^{M} W_g * S_g(v_i, v_j)) / \sum_{g=1}^{M} W_g \tag{6}$$

where W_g is the weight for algorithm g, but also the term $\frac{1}{M}$ has been replaced by a final division of $\sum_{g=1}^{M} W_g$. This is due to the fact that we normalize the matrix to a value between 0 and 1, with the maximum value that could possibly be in the matrix being the sum of all weights.

4 Experiments

This section describes the experimental setup used for the evaluation of multiple clustering algorithms, as well as the proposed PCAS and ensemble clustering methods. We perform an entire run of our entity resolution system on each data set, as explained in section 3.1. In total we perform three experiments, one for each data set. Furthermore, it should be noted that each experiment on each data set is done using seven runs, each time with a different *random state* parameter. This is done due to the fact that certain algorithms make use of random choices, as well as the machine learning models which also use a random state. By setting the random state, we ensure reproducibility, as well as getting an average result for the algorithms and models. The results are discussed per data set in section 4.4 and are reported as the average value of the seven runs.

4.1 Data sets

For our experiments, we have chosen to use the CORA citation data set, as well as the Geographical settlements and Musicbrainz data sets, which all have groundtruth labels for comparison of the final result. These data sets are all used more often in the context of entity resolution, where Geographical settlements and Musicbrainz [11] are shared to be used as benchmark for entity resolution systems.

The CORA data set is found regularly in the context of entity resolution [71, 72, 10], but not always in the same configuration. CORA consists of records on research papers in the computer science domain, which contain information like the authors, title, publisher, date and more. For our experiments we use the CORA version as used by Bilenko et al, Dong et al. and Draisbach et al., which has 1879 records that correspond to 182 different real world entities. Furthermore, we only use the columns containing the authors, title and groundtruth label. The authors column has the authors names with the main differences for duplicates being in the format in which the author names are written. Some records have the full name, others have only the first letter of the first name. More differences are found such as the use of an ampersand instead of the word "and". Additionally, the title column contains less differences per entity, with words missing or added to the title.

The Geographical settlements and Musicbrainz data sets are two of the proposed benchmark data sets by Saeedi et al. and are both also found in more entity resolution literature [11, 13, 73, 74, 75]. Geographical settlements comprises real geographic data on settlements from four knowledge bases. It contains 3054 records representing 820 real world entities and includes the groundtruth, which was obtained through manual labeling. Each record contains at least a name for the geographical settlement, but depending on the source knowledge base, it may contain extra information like the latitude, longitude and the type of settlement. For our experiments, we only use the name to match the records

on and the groundtruth label to compare our result to. As it is a data set made from four knowledge bases, the amount of records per cluster can be four at maximum. However, as we consider only dirty entity resolution without any prior knowledge on the amount of sources in the data set, we do not make use of this information.

The Musicbrainz data set is a larger data set containing 20.000 records of songs, with 10.000 entities. These 10.000 entities are from an original clean data set of songs data, which includes the title, length, artist, album, year and language of the song. A generator is used on the clean entities to generate the duplicates in the data set, but also the errors that are found in the duplicates, which can be spelling errors or empty fields. The use of a generator means that the groundtruth is automatically created, but also the amount of duplicates per record or the severeness of the errors can be used as input. For the Musicbrainz data set, the maximum amount of duplicates per record was set to 5 by Saeedi et al. [11] and the generated duplicates were made with a high degree of errors. However, for our experiment we do not take the maximum amount of duplicates per records into account. Furthermore, we only use the title, artist and album to match on, with the groundtruth label as comparison for our obtained results.

All the clustering algorithms and the unweighted ensemble clusterings are performed on each of the data sets (named *the full data sets* for clarity purposes). However, each full data set is also per run split on the connected components in a 20% test and 80% training set. This is done to allow a part of the connected components to be used with their features as training for the PCAS method. If we were to split on the full data set, it would also alter the shape of each component, and add randomness to the types of shapes in the components. Simultaneously, the training set is also used for the weighted ensemble clustering method to obtain its edges from to learn its weights. The test split is then used to cluster the clustering algorithms on, but also the PCAS method and the weighted versions of the ensemble clustering.

For the model trained by PCAS, we test the models predictions on the test data. As each prediction represents a clustering by an algorithm, we gather the clustering from the predicted label for each connected component in the test data, as well as collecting all the clusterings by the individual clustering algorithms. Simultaneously with collecting the clusterings per component, we also gather the groundtruth on a per component basis. This is done to get a gold standard clustering as proposed by Menestrina et al. [8]. By doing so we get a better idea on how well the clustering algorithms and PCAS manage to cluster on the given component, instead of punishing the algorithms for not putting nodes together that are not in the same component. Then we can evaluate the total clustering result of PCAS and compare that to the resulting clustering of the individual algorithms on the test data. Furthermore, we also evaluate the predictive capability of the model to predict the correct label for a component.

This is done by calculating an accuracy score that represents how much labels are correctly predicted as a percentage to the total amount of predictions.

For every full data set and their test split, the mixed_best theoretical limit is also included as a benchmark. This is the clustering obtained if for every component, the best clustering is chosen.

4.2 Parameters

For each experiment, we have to determine multiple parameters that we use in the entity resolution pipeline. Some parameters are different per data set used whereas others remain constant throughout all tests. Therefore, we give an overview of the consistent parameters as well as a per data set overview of the used parameters. The parameters for each individual clustering algorithm are kept consistent between different data sets. This is done so that each algorithm should create the same types of clusterings regardless of the data set used, and to prevent the tuning of many parameters for multiple algorithms. Moreover, without the tuning of parameters for the algorithms, we use the standard values given by either the authors of the algorithm, or the given implementation. Furthermore, the same values for the normalized cut are chosen for the weighted ensemble clustering and unweighted ensemble clustering, to get a comparison between the two. To get a good understanding of how well different values of the normalized cut perform, a multitude of values is used each run on the same ensemble before the final ensemble clustering. These values are shown in table 2. For hierarchical clustering, we chose to use a higher cluster threshold than the standard 0.5. This is due to the fact that we desire clusterings with a higher precision, which is achieved by hierarchical clustering by increasing the cluster threshold, at the cost of recall. For OPTICS we had to decrease the minimum amount of samples from 5 to 2. As our components are oftentimes of sizes smaller than 5, we want OPTICS to be able to create smaller clusters. By setting this value to 2, we allowed OPTICS to also decide on a clustering for components of size 3 and larger.

Parameter	**Value(s)**
Normalized cut	(0.5, 0.6, 0.7)
Hierarchical clustering - cluster_threshold	0.7
Optics - min_samples	2

Table 2: Consistent parameters used for all data sets

The biggest difference between the data sets are the blocking rules used and the Sct. The blocking rules are chosen based on the blocking rules used in other entity resolution systems that are tested on this same data set, unless the blocking rules presented by Deduplipy are superior in terms of the resulting clustering. Furthermore, multiple values for the score threshold are used, with the range decided by going above and below the best threshold found in papers using the same data set. The values for the score thresholds Sct and the blocking

rules used for CORA, Geographic settlements and Musicbrainz can be found in table 3.

Data set	**Parameter**	**Value(s)**
CORA	Sct	(0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75)
	Blocking rules	Authors - first_two_letters, Title - first_two_letters
	Similarity functions	Authors, title - adjusted_ratio & adjusted_token_sort_ratio
Geographical settlements	Sct	(0.7, 0.75, 0.8, 0.85, 0.9, 0.95)
	Blocking rules	Name - first_letter
	Similarity functions	Name - adjusted_ratio & adjusted_token_sort_ratio
Musicbrainz	Sct	(0.3, 0.35, 0.4, 0.45, 0.5, 0.55)
	Blocking rules	Album - first_letter
	Similarity functions	title, artist, album - adjusted_ratio & adjusted_token_sort_ratio

Table 3: Parameters used per data set

An active learning model is trained per data set and reused for each run on that data set. This ensures consistency in the match probabilities between runs whenever the same record pairs are presented to it. Moreover, for each data set the same similarity functions are used. This is done to demonstrate the difference in data sets and the types of strings in those data sets. Furthermore, by not using similarity functions specifically tailored to the data set, we maintain a general entity resolution system that mostly relies on the clustering to obtain a good result.

4.3 Evaluation measures

We perform an evaluation of all algorithms on the full data set, as well as an evaluation of all algorithms plus the PCAS result on the test data set. This is done using the measures proposed by Menestrina et al. [8], where they use the pairwise precision, recall, F1 and their proposed GMD method. The pairwise evaluation measures work by looking at the pairs present in the groundtruth clustering and those present in the resulting clustering. For the precision, Menestrina et al. provide equation 7. This precision can be seen as the ratio of the amount of correct pairs found in the resulting clustering to the amount of pairs found in total in the resulting clustering. The recall as described in equation 8 can be seen as the ratio of the amount of correct pairs in the resulting clustering to the amount of pairs in the groundtruth clustering.

$$PairPrecision(R, S) = \frac{|Pairs(R) \cap Pairs(S)|}{|Pairs(R)|} \tag{7}$$

$$PairRecall(R, S) = \frac{|Pairs(R) \cap Pairs(S)|}{|Pairs(S)|} \tag{8}$$

where $Pairs(R)$ and $Pairs(S)$ are the pairs present in the resulting clustering and the groundtruth clustering respectively. The pairwise F1 score is calculated in equation 9 using equations 7 & 8.

$$pF_1(R, S) = \frac{2 * PairPrecision(R, S) * PairRecall(R, S)}{PairPrecision(R, S) + PairRecall(R, S)} \tag{9}$$

Furthermore, the Variation of Information (or VoI) method by Meilă [76] is used. This is a measure similar to the GMD, where the measure looks at the difference of information lost and gained when changing one clustering to another. VoI uses the concept of entropy and mutual information, where entropy is the measure of uncertainty or randomness in a clustering. Mutual information measures the shared information between two clusterings.

The implementation of these measures is found in the python package *entity-resolution-evaluation*[12].

Furthermore, for the PCAS method, it is necessary to pick one or more evaluation measures. Therefore, the F1 score is the first metric as it is a balanced view of the precision and recall. Secondly, the GMD is used due to its close correlation to the F1 score, but also due to intuition behind it where a lower GMD also means that the clustering is closer to the ground truth. As a third option, the variation of information is used. Lastly, the precision is used before the recall is used, due to the desire to get more precise clusterings. If after going through all the metrics, there are still multiple clustering algorithms in contention (which is possible if two or more clustering algorithms come to the exact same clustering), a random algorithm out of the remaining algorithms is chosen.

[12] `https://pypi.org/project/entity-resolution-evaluation/`

4.4 Results

The experiments are performed using the methods from section 3 and the setup described in section 4.1 to section 4.3. We evaluate the implemented existing algorithms on the full data set with the different unweighted ensemble clusterings. Furthermore, we compare the existing algorithms to the (un)weighted ensemble clusterings and PCAS on the test split of the data set. The results are given on a per algorithm selection of the score threshold Sct, based on the best F1 score of that algorithm. For each table in the results, the highest values for the F1, precision and recall column are highlighted in grey. For the GMD and VoI the lowest values are highlighted in grey. A brief evaluation on the results of the full data set and the test split are also given, explaining possible reasons for the given outcomes. Appendix A contains the full resulting data per score threshold on all data sets, as well as plots of the data for visual comparison.

4.4.1 The CORA data set

Table 4 shows the results of all algorithms on the full CORA data set for their best score threshold Sct. The different ensemble clusterings are also present with an indication of the normalized cut value in the name.

algorithm	best_threshold	f1	precision	recall	GMD	VoI
mixed_best	0.500	0.894	0.863	0.927	74.857	0.320
Connected components	0.750	0.890	0.862	0.920	94.000	0.365
Hierarchical clustering	0.750	0.890	0.862	0.920	94.000	0.365
Markov clustering	0.650	0.886	0.861	0.912	86.000	0.355
OPTICS	0.650	0.324	**0.939**	0.195	510.000	1.776
IPCA	0.550	0.875	0.858	0.892	93.000	0.389
DCS	0.600	0.880	0.856	0.905	87.000	0.397
Combo	0.600	0.598	0.833	0.466	113.000	0.776
Louvain	0.500	0.622	0.840	0.494	112.000	0.739
Walktrap	0.550	0.850	0.855	0.845	88.000	0.428
Greedy modularity	0.700	0.627	0.880	0.487	128.000	0.788
DER	0.500	0.805	0.842	0.771	205.857	0.775
SCAN	0.500	0.882	0.842	**0.926**	**79.000**	0.355
Affinity propagation	0.600	0.308	0.855	0.188	365.000	1.536
ensemble_no_weight_0.5	0.550	**0.893**	0.862	**0.926**	80.000	**0.332**
ensemble_no_weight_0.6	0.500	**0.893**	0.863	0.925	80.286	0.334
ensemble_no_weight_0.7	0.650	0.838	0.854	0.823	116.143	0.530

Table 4: Results of the clustering algorithms with their best score threshold on the full CORA data set. VoI is the variation of information.

Here it is shown that there is a merit to using the ensemble clusterings, as the unweighted ensemble clusterings with a normalized cut value of 0.5 and 0.6 outperform every other individual algorithm in terms of its F1 score. Their F1 score is also only 0.1% off of the mixed_best F1 score. However, the gap to the next highest performing algorithms after the unweighted ensemble clusterings, Connected Components and Hierarchical clustering, is only 0.3%, with Markov clustering, DCS and SCAN trailing the top performers by 0.7% to 1.3%.

OPTICS has a clear lead in terms of the precision, but at the cost of recall. Furthermore, OPTICS also has a high GMD, indicating that it has created many small clusters with only certain matches in it. Interestingly, SCAN has the highest recall and lowest GMD of all the algorithms. The low GMD shows that SCAN has a clustering that requires the least amount of changes to get to the groundtruth clustering.

Table 5 shows the result of the algorithms on the test split of the CORA data set for their best score threshold. The values are generally higher on the test split compared to the evaluation on the full data set, due to the fact that the smaller amount of data to cluster on means that there are less possibilities for errors. This time, Connected components and Hierarchical clustering both have the best F1, recall, GMD and VoI. Moreover, their results are identical to that of the mixed best, with only slightly worse GMD and VoI metrics. Due to the identical results of Connected components and Hierarchical clustering, it is clear that their clusters were the exact same. As Connected components is just the component as a result, it shows that Hierarchical clustering did not actually perform any clustering as well, and kept the component as final cluster. Only OPTICS has a higher precision than any of the other algorithms, similar to the full data set evaluation. Furthermore, OPTICS has a low recall with a very high GMD, indicating that it created similar clusters as mentioned earlier.

algorithm	best_threshold	f1	precision	recall	GMD	VoI
test_split_mixed_best	0.800	0.940	0.894	1.000	3.000	0.115
test_split_connected_components	0.800	**0.940**	0.894	**1.000**	**3.429**	**0.118**
test_split_hierarchical_clustering	0.800	**0.940**	0.894	**1.000**	**3.429**	**0.118**
test_split_markov_clustering	0.550	0.934	0.885	0.996	4.714	0.158
test_split_optics	0.500	0.515	**0.965**	0.402	126.286	1.500
test_split_ipca	0.500	0.924	0.881	0.979	4.714	0.152
test_split_dcs	0.600	0.932	0.890	0.986	5.857	0.182
test_split_combo	0.700	0.686	0.876	0.571	12.857	0.533
test_split_louvain	0.700	0.687	0.874	0.573	12.857	0.531
test_split_walktrap	0.500	0.900	0.879	0.942	3.714	0.168
test_split_greedy_modularity	0.700	0.690	0.877	0.577	12.429	0.516
test_split_der	0.650	0.738	0.856	0.697	30.000	0.637
test_split_scan	0.500	0.933	0.882	0.997	3.571	0.129
test_split_affinity_propagation	0.500	0.469	0.881	0.336	51.429	1.125
test_split_ensemble_no_weight_0.5	0.600	0.936	0.892	0.994	4.857	0.156
test_split_ensemble_weighted_0.5	0.750	0.919	0.881	0.967	6.714	0.183
test_split_ensemble_no_weight_0.6	0.550	0.931	0.891	0.983	5.429	0.162
test_split_ensemble_weighted_0.6	0.750	0.919	0.881	0.967	7.143	0.188
test_split_ensemble_no_weight_0.7	0.500	0.836	0.877	0.816	7.857	0.316
test_split_ensemble_weighted_0.7	0.750	0.919	0.881	0.967	7.429	0.191
test_split_PCAS	0.800	0.935	0.894	0.991	4.571	0.141

Table 5: Results of the clustering algorithms with their best score threshold on the test split of the CORA data set. VoI is the variation of information.

The unweighted ensemble outperforms the weighted ensemble in terms of F1 on two out of the three different normalized cut values. However, the weighted

ensemble is more stable with regards to the normalized cut values. But out of the algorithms in our own methods, PCAS is the best performer, being only 0.5% behind the best performing algorithms in terms of F1 score. PCAS's GMD and VoI is also among the lowest of all algorithms, indicating that the clustering obtained by PCAS performs decent on the test set. However, when looking at all score thresholds and a selection of well performing algorithms, as seen in figure 2, we see that PCAS is often outperformed by other algorithms.

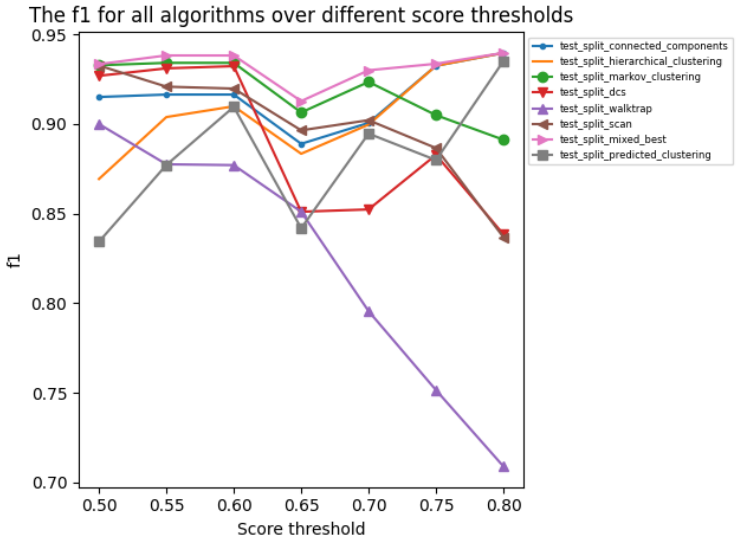

Figure 2: Selection of algorithms on the CORA data set test split for comparison of PCAS

The good performance of PCAS in the end can be denoted to the fact that PCAS noticed the good performance of Connected components and Hierarchical clustering, and has almost exclusively chosen a clustering by one of these algorithms. This effect can be seen in table 6, as the accuracy of the model is double for score threshold 0.8 compared to the other score thresholds. However, it should be noted that the accuracy score for the model is low, at points lower than randomly guessing the correct label. As the CORA data set only contains around 120 connected components, depending on the score threshold used, there are not many samples available for each different algorithm. This hurts the training of the model, which is clearly shown in table 6.

In conclusion, the results show that there are multiple algorithms capable of performing close to the theoretical mixed_best clustering, with the ensemble clustering taking the lead in terms of its F1 score. However, this indicates that more performance might be obtained by either tuning the individual algorithms to get more diverse clusterings, or by adding different algorithms. The

Score threshold	**0.50**	**0.55**	**0.60**	**0.65**	**0.70**	**0.75**	**0.80**
Accuracy score	0.071	0.063	0.08	0.074	0.063	0.066	0.153

Table 6: The PCAS predictive model average accuracy score per threshold on the CORA dataset.

test split tells a similar story, with PCAS showing that it can perform close to existing individual clustering algorithms, although outperforming clustering algorithms was the main goal. The ensembles perform well on the test split, but are bested by multiple different individual clustering algorithms. Furthermore, the weighted ensemble version shows robustness towards the normalized cut value, which could mean that different values should be used to notice the whether the normalized cut values have an impact on the weighted ensemble clustering.

4.4.2 The Geographical settlements data set

Table 7 shows the results of all algorithms on the Geographical settlements data set. The unweighted ensemble clustering with a normalized cut value of 0.6 obtained the highest F1 score with 0.661. This score is with 4% difference to the mixed_best F1 score relatively close. However, it should be noted that the GMD increases for the unweighted ensembles when the normalized cut value increases, which is a clear sign that the removal of certain nodes by cutting on the normalized cut value, generates more and smaller clusters, thus needing more merges/splits to get the groundtruth clustering. It should be noted that Markov clustering, DCS, Combo, Louvain, Walktrap and Greedy modularity, together with the other ensemble clustering algorithms are all within 2% of the aforementioned ensemble score. Hierarchical clustering has the highest precision, but lacks in its recall which reduced the F1 score to be one of the lowest. The highest recall is obtained by DCS with 0.584, which means that in the process of blocking, a lot of potential matches were already removed. Therefore using more than one blocking key might help in a scenario like this, as that creates more potential matches at the cost of computational complexity. This can be attributed to the fact that in our experiment on this data set, we did not make use of a similarity function based on the physical distance between settlements.

algorithm	**best_threshold**	**f1**	**precision**	**recall**	**GMD**	**VoI**
mixed_best	0.300	0.703	0.916	0.571	714.143	0.522
Connected components	0.400	0.571	0.668	0.499	920.000	0.694
Hierarchical clustering	0.600	0.418	**0.899**	0.273	1340.000	0.902
Markov clustering	0.300	0.644	0.834	0.525	861.000	0.624
OPTICS	0.300	0.621	0.832	0.495	1009.000	0.692
IPCA	0.300	0.535	0.860	0.388	1098.000	0.772
DCS	0.300	0.641	0.710	**0.584**	**811.000**	0.621
Combo	0.300	0.655	0.835	0.539	844.000	0.613
Louvain	0.300	0.654	0.839	0.536	847.000	0.614
Walktrap	0.300	0.646	0.773	0.555	851.000	0.628
Greedy modularity	0.300	0.654	0.840	0.535	849.000	0.616
DER	0.300	0.327	0.478	0.249	1499.714	1.060
SCAN	0.300	0.615	0.667	0.570	865.000	0.662
Affinity propagation	0.300	0.398	0.589	0.301	1438.429	0.986
ensemble_no_weight_0.5	0.300	0.652	0.763	0.569	824.000	0.614
ensemble_no_weight_0.6	0.300	**0.661**	0.816	0.555	835.000	**0.609**
ensemble_no_weight_0.7	0.300	0.649	0.877	0.515	877.571	0.625

Table 7: Results of the clustering algorithms with their best score threshold on the full Geographical settlements data set. VoI is the variation of information.

The different papers by Saeedi et al. [11, 73, 13] in which they also experiment on this data set, all calculate the distance between two settlements based on the latitude and longitude (if given). This gives more information than just the name label that we use in our experiment. As the name labels are often very short names, a small difference in the string could already result in a large dif-

ference in the similarity score which in turn results in the similarity score falling below the score threshold. An example is the geographical settlement **Petra**. Petra is found three times in the data set. Twice using the string *Petra*, once with the string *Petra (Jordan)*. The component containing all three of these occurrences is split on the fact that *Petra* and *Petra (Jordan)* are not deemed similar, which in turn results in a missed match. By using the geographical distance, more matches can be found based on the closeness of the location.

algorithm	best_threshold	f1	precision	recall	GMD	VoI
test_split_mixed_best	0.850	0.999	0.998	1.000	0.143	0.000
test_split_connected_components	0.850	0.990	0.981	1.000	2.286	0.005
test_split_hierarchical_clustering	0.850	0.990	0.981	1.000	2.286	0.005
test_split_markov_clustering	0.850	0.990	0.981	1.000	2.286	0.005
test_split_optics	0.850	0.990	0.981	1.000	2.286	0.005
test_split_ipca	0.850	0.983	0.980	0.986	3.143	0.008
test_split_dcs	0.850	0.990	0.981	1.000	2.286	0.005
test_split_combo	0.850	0.990	0.981	1.000	2.286	0.005
test_split_louvain	0.850	0.990	0.981	1.000	2.286	0.005
test_split_walktrap	0.850	0.990	0.981	1.000	2.286	0.005
test_split_greedy_modularity	0.850	0.990	0.981	1.000	2.286	0.005
test_split_der	0.350	0.471	0.659	0.381	170.143	0.538
test_split_scan	0.850	0.990	0.981	1.000	2.286	0.005
test_split_affinity_propagation	0.400	0.561	0.793	0.440	153.714	0.454
test_split_ensemble_no_weight_0.5	0.850	0.990	0.981	1.000	2.286	0.005
test_split_ensemble_weighted_0.5	0.550	0.953	0.956	0.952	12.500	0.042
test_split_ensemble_no_weight_0.6	0.850	0.990	0.981	1.000	2.286	0.005
test_split_ensemble_weighted_0.6	0.550	0.952	0.956	0.950	13.000	0.044
test_split_ensemble_no_weight_0.7	0.850	0.990	0.981	1.000	2.286	0.005
test_split_ensemble_weighted_0.7	0.600	0.927	0.945	0.912	19.143	0.059
test_split_PCAS	0.900	0.901	0.946	0.862	17.571	0.041

Table 8: Results of the clustering algorithms with their best score threshold on the test split of the Geographical settlements data set. VoI is the variation of information.

The results on the test split are shown in table 8. The scores on the test split are much higher, due to the fact that only connected components are used with the groundtruth being taken per connected component. As the input graph created on the geographical data set contains many smaller connected components, there are less cluster decisions to take by the algorithms. Therefore, a lot of the algorithms obtain the same clustering, which is not clustering on the component at all. Despite this observation, neither the weighted ensemble clusterings nor PCAS managed to get to the same results as many other algorithms. This could be due to both models not being able to learn properly with this many similar clustering results, and being punished when any other algorithm than the top performing algorithms are chosen.

Table 9 shows the accuracies of the predictive model, which shows a similar result as that for the CORA data set. For many score thresholds, the accuracy is similar or worse to randomly choosing a label, with an exception being a score

Score threshold	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
Accuracy score	0.070	0.075	0.081	0.075	0.067	0.074	0.085	0.072	0.084	0.063	0.068	0.026	0.101	0.065

Table 9: The PCAS predictive model average accuracy score per threshold on the Geographical settlements data set.

threshold of 0.9. This is also the score threshold where PCAS obtains its best average result, by better selection of the correct clustering label.

All in all, the results on the full data set as well as the test split of the data set tell us that the biggest improvement is found either by using different clustering algorithms or by changing one of the steps in the entity resolution system before the clustering part. However, the ensemble clustering again outperforms the individual clustering algorithms on the full data set, coming closest to the potential mixed_best clustering.

4.4.3 The Musicbrainz data set

Table 10 shows the results obtained by all clustering algorithms on the full Musicbrainz data set. The highest F1 score is obtained by the unweighted ensemble clustering with a normalized cut value of 0.7. This F1 score is 2.4% lower than the highest obtainable F1 score denoted by the mixed_best clustering. Furthermore, the closest individual clustering algorithms (Greedy modularity, Markov clustering, Combo and Louvain) were outperformed by between 0.7% and 0.9%. However, Markov clustering had the lowest GMD, even lower than that of the best unweighted ensemble clustering. And not just the F1 score of the unweighted ensemble clustering was the highest, also the precision of the unweighted ensemble clustering outperformed the other algorithms, this time differing 4.5% compared to the mixed_best clustering. The recall of all the algorithms is not very high, again showing possible signs that the blocking performed was too restrictive, putting actual matches in different connected components.

algorithm	best_threshold	f1	precision	recall	GMD	VoI
mixed_best	0.300	0.693	0.957	0.543	3596.143	0.350
Connected components	0.550	0.629	0.862	0.496	4152.000	0.407
Hierarchical clustering	0.550	0.571	0.860	0.428	4774.000	0.460
Markov clustering	0.350	0.661	0.871	0.533	**3886.000**	0.383
OPTICS	0.350	0.655	0.895	0.516	4089.000	0.396
IPCA	0.300	0.629	0.905	0.482	4198.000	0.411
DCS	0.400	0.651	0.841	0.531	3932.000	0.391
Combo	0.350	0.661	0.875	0.532	3933.000	0.385
Louvain	0.350	0.660	0.869	0.532	3939.000	0.386
Walktrap	0.350	0.658	0.859	0.534	3929.000	0.387
Greedy modularity	0.350	0.662	0.875	0.532	3924.000	0.385
DER	0.500	0.249	0.760	0.149	7408.857	0.691
SCAN	0.500	0.644	0.884	0.507	4068.000	0.397
Affinity propagation	0.450	0.276	0.758	0.169	7562.286	0.693
ensemble_no_weight_0.5	0.350	0.652	0.828	**0.538**	3915.571	0.390
ensemble_no_weight_0.6	0.400	0.660	0.878	0.528	3902.857	0.384
ensemble_no_weight_0.7	0.350	**0.669**	**0.912**	0.528	3917.714	**0.380**

Table 10: Results of the clustering algorithms with their best score threshold on the full Musicbrainz data set. VoI is the variation of information.

The test split results are presented in table 11. Similar to the results on the full data set, the highest F1 score is obtained by the unweighted ensemble clustering with a normalized cut value of 0.7. However, Combo and Louvain obtain the exact same F1 score of 0.966. In terms of precision, the ensemble clustering has highest value, closely followed by the other two top performing algorithms. Interestingly, the GMD and the VoI of Markov clustering are the best out of all the algorithms. Although Markov clustering does not have the highest F1 score, having the lowest GMD does indicate that it came to a clustering that is on average the closest to the groundtruth clustering.

The unweighted ensemble clusterings with different normalized cut values all outperform the weighted ensemble clustering in terms of their F1 score.

algorithm	best_threshold	f1	precision	recall	GMD	VoI
test_split_mixed_best	0.550	0.984	0.971	0.998	17.714	0.010
test_split_connected_components	0.550	0.938	0.885	**1.000**	39.143	0.027
test_split_hierarchical_clustering	0.550	0.872	0.885	0.861	163.286	0.080
test_split_markov_clustering	0.550	0.962	0.930	0.996	**37.857**	**0.022**
test_split_optics	0.550	0.945	0.945	0.946	75.143	0.038
test_split_ipca	0.350	0.907	0.933	0.882	133.857	0.069
test_split_dcs	0.550	0.956	0.919	0.997	41.000	0.025
test_split_combo	0.550	**0.966**	0.953	0.980	45.571	0.023
test_split_louvain	0.550	**0.966**	0.953	0.980	46.143	0.024
test_split_walktrap	0.550	0.965	0.944	0.987	43.429	0.023
test_split_greedy_modularity	0.550	0.964	0.948	0.980	46.857	0.024
test_split_der	0.500	0.431	0.831	0.291	720.143	0.325
test_split_scan	0.550	0.958	0.925	0.994	42.000	0.024
test_split_affinity_propagation	0.500	0.469	0.856	0.324	745.429	0.323
test_split_ensemble_no_weight_0.5	0.550	0.959	0.924	0.998	38.143	0.023
test_split_ensemble_weighted_0.5	0.550	0.852	0.915	0.851	178.571	0.081
test_split_ensemble_no_weight_0.6	0.550	0.964	0.936	0.993	39.286	**0.022**
test_split_ensemble_weighted_0.6	0.550	0.848	0.915	0.846	181.571	0.082
test_split_ensemble_no_weight_0.7	0.550	**0.966**	**0.955**	0.977	47.429	0.024
test_split_ensemble_weighted_0.7	0.550	0.847	0.917	0.841	184.143	0.083
test_split_predicted_PCAS	0.350	0.891	0.919	0.865	221.143	0.097

Table 11: Results of the clustering algorithms with their best score threshold on the test split of the Musicbrainz data set. VoI is the variation of information.

Furthermore, the weighted ensemble clusterings all have a relatively high GMD compared to the individual clustering algorithms, only having a better GMD than the bottom two algorithms. The GMD by PCAS is even worse, which can be explained by the fact that the model accuracy is again similar or worse than randomly predicting the labels, as seen in table 12. This results in PCAS also picking clusterings obtained by the bottom performers of the individual clustering algorithms. Table 12 shows that PCAS has the highest accuracy score at score threshold 0.35, which is also the score threshold for which PCAS obtains its highest F1 score, as per table 11.

Score threshold	**0.30**	**0.35**	**0.40**	**0.45**	**0.50**	**0.55**
Accuracy score	0.076	0.088	0.084	0.079	0.061	0.069

Table 12: The PCAS predictive model average accuracy score per threshold on the Musicbrainz dataset.

In summary, the unweighted ensemble clustering has one off the best performing clustering result on multiple metrics. Furthermore, PCAS showed that without proper predictions, its result will fall below most individual algorithms.

5 Conclusion & future work

In this paper we have presented a comparative evaluation of multiple clustering algorithms not frequently used for entity resolution, as well as known clustering algorithms in the context of entity resolution, using a full entity resolution pipeline. The evaluation was done over multiple different data sets containing the groundtruth label and are therefore used as benchmark for entity resolution systems. For the evaluation we have looked at multiple metrics often used in clustering and entity resolution, creating a comparitive overview of many different algorithms. In our experiments, we have not been able to appoint a single 'goto' algorithm that would always perform the best. However, in our evaluation there are multiple top performing algorithms, with the likes of Markov clustering, SCAN, DCS and Greedy modularity providing satisfactory results.

Furthermore, we have introduced and implemented a predictive model (PCAS) in section 3.2 that predicts which clustering algorithm to use based on features of a given component. This model is trained and evaluated for every data set, as well as compared to the other used clustering algorithms. We have demonstrated that PCAS does not perform up to the standard of other algorithms yet, but does show its potential on different occasions.

Lastly we have implemented and evaluated a weighted and unweighted version of ensemble clustering in section 3.3. Similar to PCAS, it makes use of other clustering algorithms to get to a final clustering. The results show that the implementation of unweighted ensemble clustering based on the implementation by Yu et al. [70] provides satisfactory results on occasion, outperforming individual clustering algorithms on every data set. Although our proposed implementation of a weighted ensemble clustering did perform close to the unweighted version, it rarely outperformed the unweighted ensembles or most individual clustering algorithms.

Limitations

This paper has potential limitations. By not including the exact setup for the entity resolution system, the given results can not be compared one on one to the results from previous studies on the same data sets [10, 11]. However, as this study has included algorithms that are also used in previous studies, the comparison could be made with the performance relative to that of the overlapping algorithm(s).

Furthermore, this study did not perform any tuning of the parameters of the individual algorithms. This may have resulted in worse performance compared to the potential performance of each individual algorithm. But not only the per algorithm parameters were not tuned, those of the predictive PCAS model

and the linear regression used to learn the weights for weighted ensemble clustering were also not tuned. This could have increased the performance of the models resulting in better final clusterings. Using more edges as training for the weighted ensemble linear regression model could have lead to better tuned weights, which in turn could result in better clusterings.

Moreover, a selection of well performing algorithms on the data sets could have been used to explore the impact on PCAS and the ensemble clustering.

Finally, only a single type of ensemble clustering was used, namely that of Yu et al. However, many more ensemble clustering methods exist, which could have been used as a more comparative measure of the ensemble clustering quality.

Future work

For future work, the potential performance of PCAS could be increased to obtain better predictions (and thus better clusterings). This could for example be done through adding more descriptive features when training the model. Finding more descriptive features could emphasise the difference in clusterings made by different clustering algorithms, allowing the model to better learn what type of component each clustering algorithm would perform well on. Furthermore, filtering the used algorithms by best performing algorithms to choose from could increase the performance of PCAS. As in that case it would disable the possibility for the model to predict a bad performing clustering algorithm as the label for a component. This could lead to an interesting evaluation of which algorithms to allow as input and how many algorithms to pick.

Moreover, in a future research with a better performing PCAS model, it could be used to predict the weights for the weighted ensemble clustering. That would fit into the same intuition where on a per component basis the best performing algorithm would get a higher weight and thus be more important in the consensus of the ensemble.

References

[1] Stephen Kaisler et al. “Big data: Issues and challenges moving forward”. In: *2013 46th Hawaii international conference on system sciences*. IEEE. 2013, pp. 995–1004.

[2] Maik Fröbe et al. “Sampling bias due to near-duplicates in learning to rank”. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020, pp. 1997–2000.

[3] Christopher W Kelman, A John Bass, and Cashel DJ Holman. “Research use of linked health data—a best practice protocol”. In: *Australian and New Zealand journal of public health* 26.3 (2002), pp. 251–255.

[4] HB Newcombe et al. “Automatic linkage of vital records.” In: (1967).

[5] Ivan P Fellegi and Alan B Sunter. “A theory for record linkage”. In: *Journal of the American Statistical Association* 64.328 (1969), pp. 1183–1210.

[6] Thilina Ranbaduge, Peter Christen, and Rainer Schnell. “Large Scale Record Linkage in the Presence of Missing Data”. In: *arXiv preprint arXiv:2104.09677* (2021).

[7] Y Richard Wang and Stuart E Madnick. “The Inter-Database Instance Identification Problem in Integrating Autonomous Systems.” In: *ICDE*. 1989, pp. 46–55.

[8] David Menestrina, Steven Euijong Whang, and Hector Garcia-Molina. “Evaluating entity resolution results”. In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 208–219.

[9] Oktie Hassanzadeh et al. “Framework for evaluating clustering algorithms in duplicate detection”. In: *Proceedings of the VLDB Endowment* 2.1 (2009), pp. 1282–1293.

[10] Uwe Draisbach, Peter Christen, and Felix Naumann. “Transforming pairwise duplicates to entity clusters for high-quality duplicate detection”. In: *Journal of Data and Information Quality (JDIQ)* 12.1 (2019), pp. 1–30.

[11] 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*. Springer. 2017, pp. 278–293.

[12] Islam Akef Ebeid, John R Talburt, and Md Abdus Salam Siddique. “Graph-based hierarchical record clustering for unsupervised entity resolution”. In: *ITNG 2022 19th International Conference on Information Technology-New Generations*. Springer. 2022, pp. 107–118.

[13] Alieh Saeedi, Eric Peukert, and Erhard Rahm. “Using link features for entity clustering in knowledge graphs”. In: *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*. Springer. 2018, pp. 576–592.

[14] Syed Agha Muhammad and Kristof Van Laerhove. "DCS: Divide and Conquer Strategy for Detecting Overlapping Communities in Social Graphs". unpublished.

[15] Alexander Topchy, Anil K Jain, and William Punch. "Clustering ensembles: Models of consensus and weak partitions". In: *IEEE transactions on pattern analysis and machine intelligence* 27.12 (2005), pp. 1866–1881.

[16] Zhaoqi Chen, Dmitri V Kalashnikov, and Sharad Mehrotra. "Exploiting context analysis for combining multiple entity resolution systems". In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 2009, pp. 207–218.

[17] Vassilis Christophides et al. "An overview of end-to-end entity resolution for big data". In: *ACM Computing Surveys (CSUR)* 53.6 (2020), pp. 1–42.

[18] Peter Christen. "A survey of indexing techniques for scalable record linkage and deduplication". In: *IEEE transactions on knowledge and data engineering* 24.9 (2011), pp. 1537–1555.

[19] George Papadakis et al. "Efficient entity resolution for large heterogeneous information spaces". In: *Proceedings of the fourth ACM international conference on Web search and data mining*. 2011, pp. 535–544.

[20] George Papadakis et al. "A blocking framework for entity resolution in highly heterogeneous information spaces". In: *IEEE Transactions on Knowledge and Data Engineering* 25.12 (2012), pp. 2665–2682.

[21] George Papadakis et al. "Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data". In: *Proceedings of the fifth ACM international conference on Web search and data mining*. 2012, pp. 53–62.

[22] George Papadakis et al. "Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data". In: *Proceedings of the VLDB Endowment* 9.4 (2015), pp. 312–323.

[23] Mauricio A Hernández and Salvatore J Stolfo. "Real-world data is dirty: Data cleansing and the merge/purge problem". In: *Data mining and knowledge discovery* 2.1 (1998), pp. 9–37.

[24] Vladimir I Levenshtein et al. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Vol. 10. 8. Soviet Union. 1966, pp. 707–710.

[25] William E Winkler. "String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage." In: (1990).

[26] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. "Duplicate record detection: A survey". In: *IEEE Transactions on knowledge and data engineering* 19.1 (2006), pp. 1–16.

[27] S Fortunato. "Community detection in graphs/Santo Fortunato". In: *Physics Reports* (2009).

[28] Ryan McConville, Weiru Liu, and Jun Hong. "Vertex Deduplication Based on String Similarity and Community Membership". In: *Complex Networks & Their Applications VI: Proceedings of Complex Networks 2017 (The Sixth International Conference on Complex Networks and Their Applications)*. Springer. 2018, pp. 178–189.

[29] John Hopcroft and Robert Tarjan. "Algorithm 447: efficient algorithms for graph manipulation". In: *Communications of the ACM* 16.6 (1973), pp. 372–378.

[30] Robin Sibson. "SLINK: an optimally efficient algorithm for the single-link cluster method". In: *The computer journal* 16.1 (1973), pp. 30–34.

[31] Angur Mahmud Jarman. "Hierarchical cluster analysis: Comparison of single linkage, complete linkage, average linkage and centroid linkage method". In: *Georgia Southern University* (2020).

[32] Alieh Saeedi, Lucie David, and Erhard Rahm. "Matching Entities from Multiple Sources with Hierarchical Agglomerative Clustering." In: *KEOD*. 2021, pp. 40–50.

[33] Wanying Xu et al. "Unsupervised Entity Resolution Method Based on Random Forest". In: *International Conference on Web Information Systems and Applications*. Springer. 2021, pp. 372–382.

[34] Phuc Quang Tran et al. "An Affinity Propagation Approach for Entity Clustering with Spark". In: *Proceedings of the 2020 5th International Conference on Intelligent Information Technology*. 2020, pp. 51–55.

[35] Stefan Lerm, Alieh Saeedi, and Erhard Rahm. "Extended affinity propagation clustering for multi-source entity resolution". In: *BTW 2021* (2021).

[36] Brendan J Frey and Delbert Dueck. "Clustering by passing messages between data points". In: *science* 315.5814 (2007), pp. 972–976.

[37] Mihael Ankerst et al. "OPTICS: Ordering points to identify the clustering structure". In: *ACM Sigmod record* 28.2 (1999), pp. 49–60.

[38] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[39] Mark EJ Newman and Michelle Girvan. "Finding and evaluating community structure in networks". In: *Physical review E* 69.2 (2004), p. 026113.

[40] Sanjeev Shenoy et al. "Deduplication in a massive clinical note dataset". In: *arXiv preprint arXiv:1704.05617* (2017).

[41] Stanislav Sobolevsky et al. "General optimization technique for high-quality community detection in complex networks". In: *Physical Review E* 90.1 (2014), p. 012811.

[42] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. "Finding community structure in very large networks". In: *Physical review E* 70.6 (2004), p. 066111.

[43] Stijn Marinus Van Dongen. "Graph clustering by flow simulation". PhD thesis. 2000.

[44] Pascal Pons and Matthieu Latapy. "Computing communities in large networks using random walks". In: *Computer and Information Sciences-ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20*. Springer. 2005, pp. 284–293.

[45] Mark Kozdoba and Shie Mannor. "Community detection via measure space embedding". In: *Advances in neural information processing systems* 28 (2015).

[46] Xiaowei Xu et al. "Scan: a structural clustering algorithm for networks". In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007, pp. 824–833.

[47] Min Li et al. "Modifying the DPClus algorithm for identifying protein complexes based on new topological structures". In: *BMC bioinformatics* 9.1 (2008), pp. 1–16.

[48] Keyvan Golalipour et al. "From clustering to clustering ensemble selection: A review". In: *Engineering Applications of Artificial Intelligence* 104 (2021), p. 104388.

[49] Jukka-Pekka Onnela et al. "Intensity and coherence of motifs in weighted complex networks". In: *Physical Review E* 71.6 (2005), p. 065103.

[50] Nykamp DQ. *Definition of the transitivity of a graph*. URL: `http://mathinsight.org/definition/transitivity_graph` (visited on 04/15/2023).

[51] Paolo Boldi and Sebastiano Vigna. "Axioms for centrality". In: *Internet Mathematics* 10.3-4 (2014), pp. 222–262.

[52] Tom M. Mitchell. *Machine Learning*. McGraw Hill, 1997. ISBN: 0070428077.

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[54] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[55] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[56] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*. Vol. 398. John Wiley & Sons, 2013.

[57] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[58] Burr Settles. "Active learning literature survey". In: (2009).

[59] David D Lewis. "A sequential algorithm for training text classifiers: Corrigendum and additional data". In: *Acm Sigir Forum*. Vol. 29. 2. ACM New York, NY, USA. 1995, pp. 13–19.

[60] Vikas Yadav and Steven Bethard. "A survey on recent advances in named entity recognition from deep learning models". In: *arXiv preprint arXiv:1910.11470* (2019).

[61] Sunita Sarawagi and Anuradha Bhamidipaty. "Interactive deduplication using active learning". In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, pp. 269–278.

[62] Peter Christen. "Probabilistic data generation for deduplication and data linkage". In: *Intelligent Data Engineering and Automated Learning-IDEAL 2005: 6th International Conference, Brisbane, Australia, July 6-8, 2005. Proceedings 6*. Springer. 2005, pp. 109–116.

[63] Hanna Köpcke, Andreas Thor, and Erhard Rahm. "Evaluation of entity resolution approaches on real-world match problems". In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 484–493.

[64] Kai Hildebrandt et al. "Large-scale data pollution with Apache Spark". In: *IEEE Transactions on Big Data* 6.2 (2017), pp. 396–411.

[65] Alexander Strehl and Joydeep Ghosh. "Cluster ensembles—a knowledge reuse framework for combining multiple partitions". In: *Journal of machine learning research* 3.Dec (2002), pp. 583–617.

[66] Bo Long, Zhongfei Zhang, and Philip S Yu. "Combining multiple clusterings by soft correspondence". In: *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE. 2005, 8–pp.

[67] Giulio Rossetti, Letizia Milli, and Rémy Cazabet. "CDLIB: a python library to extract, compare and evaluate communities from complex networks". In: *Applied Network Science* 4.1 (2019), pp. 1–26.

[68] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[69] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[70] Zhiwen Yu, Hau-San Wong, and Hongqiang Wang. "Graph-based consensus clustering for class discovery from gene expression data". In: *Bioinformatics* 23.21 (2007), pp. 2888–2896.

[71] Mikhail Bilenko and Raymond J Mooney. "Adaptive duplicate detection using learnable string similarity measures". In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003, pp. 39–48.

[72] Xin Dong, Alon Halevy, and Jayant Madhavan. "Reference reconciliation in complex information spaces". In: *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 2005, pp. 85–96.

[73] Alieh Saeedi et al. “Scalable matching and clustering of entities with FAMER”. In: *Complex Systems Informatics and Modeling Quarterly* 16 (2018), pp. 61–83.

[74] Markus Nentwig et al. “Distributed Holistic Clustering on Linked Data”. In: *On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part II*. 2017, pp. 371–382. DOI: 10.1007/978-3-319-69459-7_25. URL: https://doi.org/10.1007/978-3-319-69459-7_25.

[75] Markus Nentwig and Erhard Rahm. “Incremental clustering on linked data”. In: *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2018, pp. 531–538.

[76] Marina Meilă. “Comparing clusterings by the variation of information”. In: *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*. Springer. 2003, pp. 173–187.

A Appendix - Full results

A.1 CORA

Algorithm	0.50	0.55	0.60	0.65	0.70	0.75	0.80
mixed_best	0.894	0.894	0.893	0.893	0.893	0.891	0.874
connected_components	0.838	0.856	0.856	0.859	0.881	0.890	0.874
hierarchical_clustering	0.814	0.849	0.845	0.849	0.879	0.890	0.874
markov_clustering	0.882	0.882	0.882	0.886	0.876	0.863	0.837
optics	0.199	0.218	0.294	0.324	0.274	0.200	0.201
cdlib_ipca	0.872	0.875	0.867	0.860	0.823	0.778	0.692
cdlib_dcs	0.874	0.879	0.880	0.837	0.839	0.843	0.732
cdlib_combo	0.571	0.566	0.598	0.566	0.594	0.501	0.538
louvain	0.622	0.517	0.596	0.582	0.598	0.495	0.510
walktrap	0.849	0.850	0.850	0.845	0.769	0.711	0.666
greedy_modularity	0.559	0.566	0.568	0.566	0.627	0.617	0.572
cdlib_der	0.805	0.702	0.691	0.730	0.723	0.645	0.682
cdlib_scan	0.882	0.873	0.872	0.874	0.869	0.843	0.780
affinity_propagation	0.295	0.284	0.308	0.296	0.283	0.224	0.245
ensemble_no_weight_0.5	0.887	0.893	0.889	0.879	0.866	0.867	0.846
ensemble_no_weight_0.6	0.893	0.890	0.888	0.866	0.864	0.842	0.805
ensemble_no_weight_0.7	0.791	0.793	0.758	0.838	0.787	0.610	0.648

Table 13: Results of the F1 metric of the clustering algorithms for each threshold on the full data of the CORA data set.

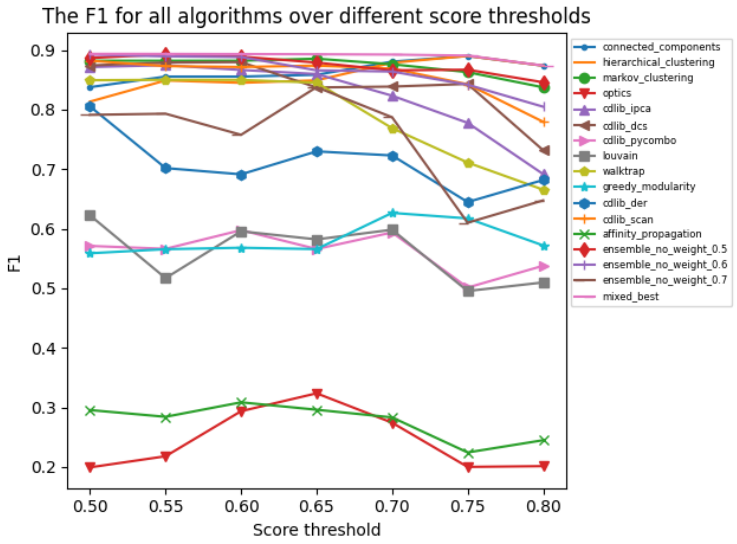

Figure 3: All algorithms on the CORA data set.

Algorithm	0.50	0.55	0.60	0.65	0.70	0.75	0.80
mixed_best	0.863	0.863	0.863	0.863	0.865	0.863	0.859
connected_components	0.761	0.791	0.791	0.798	0.842	0.862	0.859
hierarchical_clustering	0.769	0.806	0.788	0.794	0.842	0.862	0.859
markov_clustering	0.842	0.842	0.842	0.861	0.859	0.856	0.860
optics	0.956	0.942	0.820	0.939	0.964	0.937	0.950
cdlib_ipca	0.841	0.858	0.858	0.859	0.859	0.867	0.895
cdlib_dcs	0.858	0.851	0.856	0.834	0.863	0.886	0.840
cdlib_combo	0.825	0.827	0.833	0.845	0.854	0.867	0.866
louvain	0.840	0.875	0.833	0.849	0.845	0.873	0.882
walktrap	0.842	0.855	0.854	0.854	0.872	0.913	0.911
greedy_modularity	0.820	0.828	0.867	0.836	0.880	0.862	0.876
cdlib_der	0.842	0.818	0.815	0.828	0.881	0.869	0.870
cdlib_scan	0.842	0.841	0.840	0.859	0.860	0.861	0.867
affinity_propagation	0.888	0.854	0.855	0.856	0.912	0.914	0.907
ensemble_no_weight_0.5	0.851	0.862	0.861	0.859	0.856	0.858	0.857
ensemble_no_weight_0.6	0.863	0.861	0.861	0.856	0.857	0.854	0.857
ensemble_no_weight_0.7	0.880	0.882	0.835	0.854	0.847	0.861	0.886

Table 14: Results of the precision metric of the clustering algorithms for each threshold on the full data of the CORA data set.

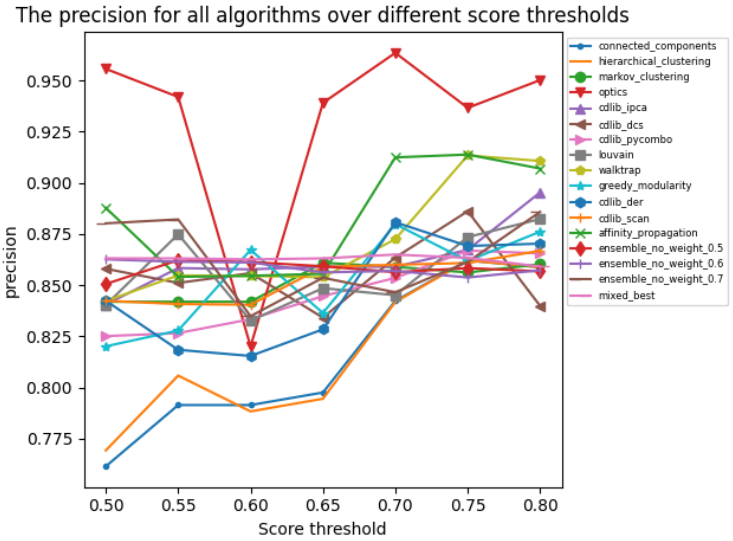

Figure 4: All algorithms on the CORA data set.

Algorithm	0.50	0.55	0.60	0.65	0.70	0.75	0.80
mixed_best	0.927	0.926	0.926	0.925	0.922	0.920	0.890
connected_components	0.931	0.931	0.931	0.930	0.923	0.920	0.890
hierarchical_clustering	0.863	0.897	0.911	0.912	0.919	0.920	0.890
markov_clustering	0.927	0.927	0.927	0.912	0.894	0.870	0.816
optics	0.111	0.123	0.179	0.195	0.160	0.112	0.113
cdlib_ipca	0.905	0.892	0.876	0.861	0.791	0.705	0.564
cdlib_dcs	0.891	0.909	0.905	0.841	0.815	0.804	0.648
cdlib_combo	0.437	0.431	0.466	0.425	0.455	0.352	0.390
louvain	0.494	0.367	0.464	0.443	0.463	0.346	0.358
walktrap	0.857	0.845	0.846	0.837	0.687	0.582	0.524
greedy_modularity	0.424	0.430	0.422	0.427	0.487	0.481	0.425
cdlib_der	0.771	0.615	0.600	0.652	0.614	0.514	0.563
cdlib_scan	0.926	0.908	0.905	0.890	0.878	0.825	0.708
affinity_propagation	0.177	0.170	0.188	0.179	0.168	0.128	0.142
ensemble_no_weight_0.5	0.926	0.926	0.919	0.900	0.876	0.876	0.835
ensemble_no_weight_0.6	0.925	0.920	0.918	0.876	0.871	0.830	0.759
ensemble_no_weight_0.7	0.719	0.721	0.694	0.823	0.736	0.474	0.510

Table 15: Results of the recall metric of the clustering algorithms for each threshold on the full data of the CORA data set.

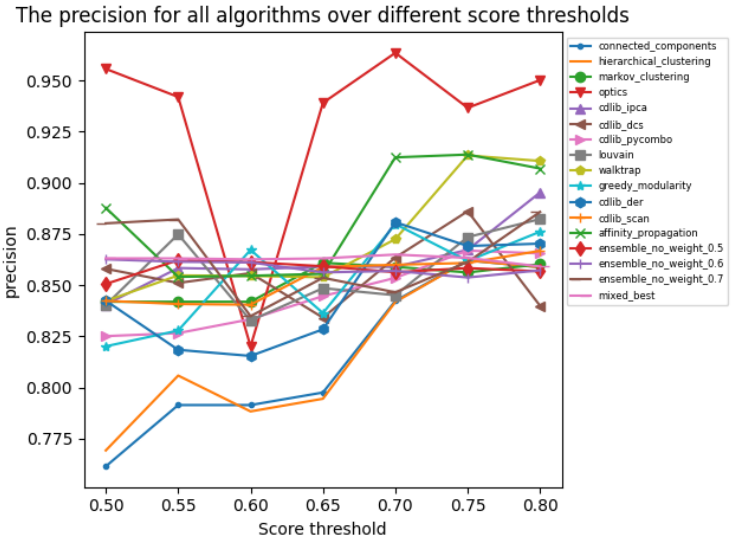

Figure 5: All algorithms on the CORA data set.

Algorithm	0.50	0.55	0.60	0.65	0.70	0.75	0.80
mixed_best	74.857	76.857	78.143	82.000	87.286	90.143	117.857
connected_components	79.000	79.000	79.000	84.000	91.000	94.000	121.000
hierarchical_clustering	110.000	102.000	90.000	88.000	92.000	94.000	121.000
markov_clustering	79.000	80.000	80.000	86.000	93.000	101.000	142.000
optics	806.000	703.000	567.000	510.000	399.000	426.000	419.000
cdlib_ipca	89.000	93.000	106.000	118.000	130.000	156.000	185.000
cdlib_dcs	81.000	82.000	87.000	93.000	107.000	111.000	148.000
cdlib_combo	112.000	112.000	113.000	118.000	130.000	151.000	183.000
louvain	112.000	115.000	113.000	118.000	130.000	153.000	187.000
walktrap	83.000	88.000	97.000	101.000	127.000	153.000	210.000
greedy_modularity	115.000	113.000	113.000	119.000	128.000	146.000	177.000
cdlib_der	205.857	205.429	205.429	212.714	223.429	230.286	270.286
cdlib_scan	79.000	86.000	90.000	93.000	104.000	125.000	165.000
affinity_propagation	370.429	388.571	365.000	397.000	439.571	508.286	547.286
ensemble_no_weight_0.5	79.571	80.000	81.000	88.000	93.714	101.143	140.857
ensemble_no_weight_0.6	80.286	82.286	83.000	92.429	101.571	111.000	154.000
ensemble_no_weight_0.7	105.000	109.143	114.429	116.143	136.286	183.429	208.286

Table 16: Results of the GMD metric of the clustering algorithms for each threshold on the full data of the CORA data set.

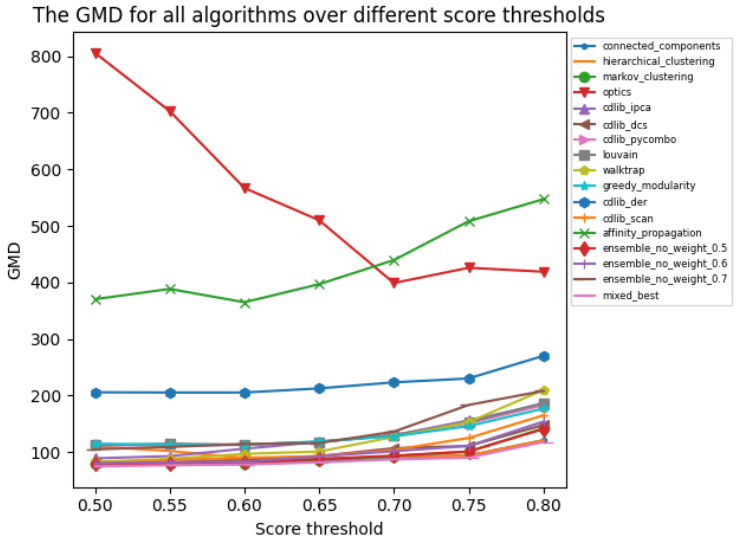

Figure 6: All algorithms on the CORA data set.

Algorithm	0.50	0.55	0.60	0.65	0.70	0.75	0.80
mixed_best	0.320	0.323	0.326	0.333	0.345	0.355	0.440
connected_components	0.413	0.388	0.388	0.395	0.381	0.365	0.445
hierarchical_clustering	0.504	0.433	0.420	0.413	0.384	0.365	0.445
markov_clustering	0.356	0.358	0.358	0.355	0.381	0.432	0.571
optics	2.233	2.098	1.870	1.776	1.681	1.857	1.852
cdlib_ipca	0.395	0.389	0.434	0.472	0.550	0.702	0.898
cdlib_dcs	0.390	0.374	0.397	0.466	0.482	0.488	0.700
cdlib_combo	0.780	0.788	0.776	0.801	0.817	0.979	1.014
louvain	0.739	0.852	0.779	0.800	0.810	0.987	1.062
walktrap	0.422	0.428	0.463	0.488	0.636	0.809	0.986
greedy_modularity	0.806	0.799	0.803	0.805	0.788	0.854	0.970
cdlib_der	0.775	0.822	0.819	0.805	0.855	0.942	0.985
cdlib_scan	0.355	0.383	0.395	0.381	0.418	0.526	0.713
affinity_propagation	1.560	1.603	1.536	1.593	1.684	1.873	1.883
ensemble_no_weight_0.5	0.347	0.332	0.349	0.380	0.390	0.440	0.561
ensemble_no_weight_0.6	0.334	0.343	0.357	0.417	0.414	0.508	0.674
ensemble_no_weight_0.7	0.575	0.582	0.616	0.530	0.643	0.942	1.019

Table 17: Results of the VoI metric of the clustering algorithms for each threshold on the full data of the CORA data set.

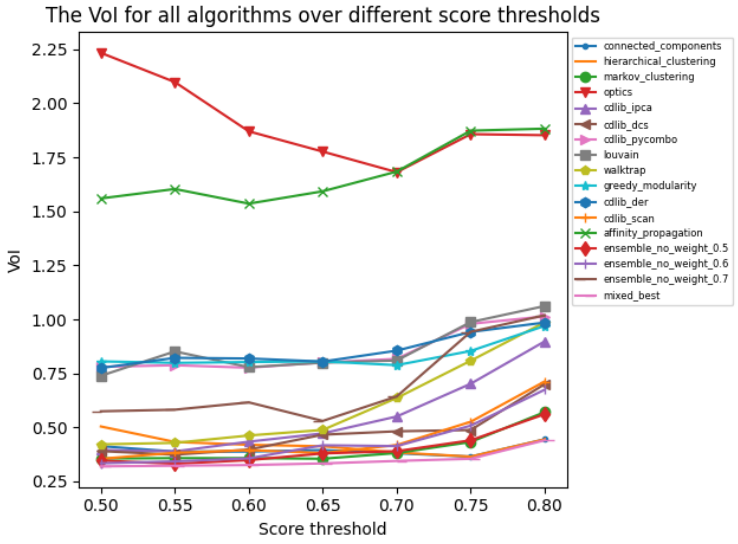

Figure 7: All algorithms on the CORA data set.

CORA - Test split

Algorithm	0.50	0.55	0.60	0.65	0.70	0.75	0.80
test_split_mixed_best	0.933	0.938	0.938	0.913	0.930	0.934	0.940
test_split_connected_components	0.915	0.916	0.916	0.889	0.901	0.932	0.940
test_split_hierarchical_clustering	0.869	0.904	0.910	0.883	0.900	0.932	0.940
test_split_markov_clustering	0.933	0.934	0.934	0.906	0.923	0.905	0.891
test_split_optics	0.515	0.379	0.445	0.328	0.390	0.323	0.413
test_split_ipca	0.924	0.911	0.901	0.870	0.836	0.776	0.741
test_split_dcs	0.927	0.931	0.932	0.851	0.852	0.883	0.838
test_split_combo	0.658	0.647	0.666	0.622	0.686	0.584	0.639
test_split_louvain	0.674	0.614	0.664	0.633	0.687	0.578	0.618
test_split_walktrap	0.900	0.878	0.877	0.851	0.796	0.751	0.709
test_split_greedy_modularity	0.643	0.649	0.649	0.622	0.690	0.645	0.646
test_split_der	0.692	0.699	0.708	0.738	0.691	0.640	0.677
test_split_scan	0.933	0.921	0.920	0.896	0.902	0.887	0.837
test_split_affinity_propagation	0.469	0.410	0.429	0.356	0.420	0.321	0.415
test_split_ensemble_no_weight_0.5	0.933	0.932	0.936	0.888	0.913	0.886	0.876
test_split_ensemble_weighted_0.5	0.779	0.615	0.810	0.873	0.887	0.919	0.862
test_split_ensemble_no_weight_0.6	0.927	0.931	0.929	0.875	0.909	0.864	0.801
test_split_ensemble_weighted_0.6	0.779	0.547	0.810	0.873	0.887	0.919	0.792
test_split_ensemble_no_weight_0.7	0.836	0.802	0.789	0.828	0.767	0.641	0.669
test_split_ensemble_weighted_0.7	0.779	0.496	0.810	0.873	0.887	0.919	0.745
test_split_predicted_clustering	0.834	0.877	0.910	0.842	0.895	0.880	0.935

Table 18: Results of the F1 metric of the clustering algorithms for each threshold on the test split of the CORA data set.

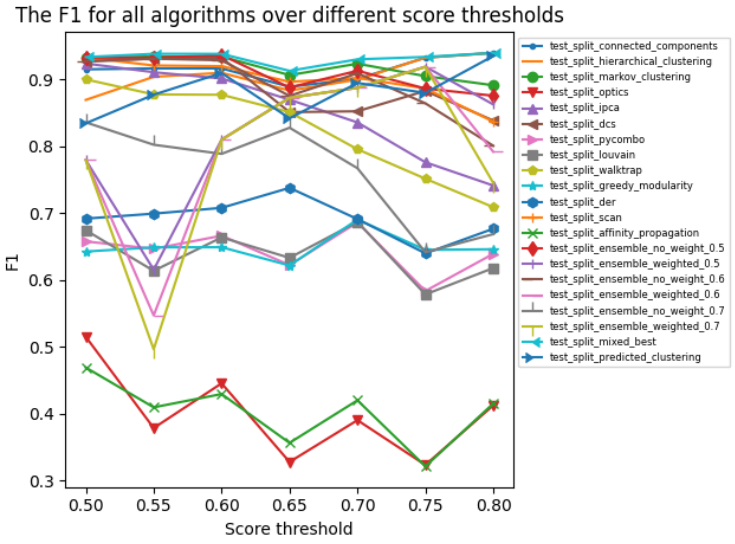

Figure 8: All algorithms on the CORA data set test split.

Algorithm	0.50	0.55	0.60	0.65	0.70	0.75	0.80
test_split_mixed_best	0.884	0.893	0.893	0.856	0.872	0.884	0.894
test_split_connected_components	0.854	0.859	0.859	0.822	0.827	0.882	0.894
test_split_hierarchical_clustering	0.851	0.864	0.858	0.821	0.827	0.882	0.894
test_split_markov_clustering	0.882	0.885	0.885	0.856	0.867	0.878	0.892
test_split_optics	0.965	0.943	0.895	0.902	0.954	0.924	0.938
test_split_ipca	0.881	0.888	0.888	0.858	0.869	0.875	0.893
test_split_dcs	0.885	0.888	0.890	0.845	0.869	0.887	0.891
test_split_combo	0.876	0.873	0.874	0.855	0.876	0.886	0.895
test_split_louvain	0.876	0.895	0.874	0.854	0.874	0.890	0.896
test_split_walktrap	0.879	0.880	0.882	0.853	0.873	0.883	0.898
test_split_greedy_modularity	0.874	0.873	0.895	0.852	0.877	0.888	0.896
test_split_der	0.878	0.881	0.885	0.856	0.866	0.882	0.896
test_split_scan	0.882	0.883	0.882	0.855	0.867	0.881	0.891
test_split_affinity_propagation	0.881	0.878	0.874	0.860	0.872	0.899	0.910
test_split_ensemble_no_weight_0.5	0.882	0.891	0.892	0.855	0.866	0.881	0.899
test_split_ensemble_weighted_0.5	0.881	0.898	0.850	0.844	0.829	0.881	0.895
test_split_ensemble_no_weight_0.6	0.881	0.891	0.891	0.853	0.873	0.881	0.898
test_split_ensemble_weighted_0.6	0.881	0.904	0.850	0.844	0.829	0.881	0.894
test_split_ensemble_no_weight_0.7	0.877	0.875	0.868	0.851	0.876	0.888	0.896
test_split_ensemble_weighted_0.7	0.881	0.901	0.849	0.844	0.829	0.881	0.891
test_split_predicted_clustering	0.883	0.856	0.863	0.820	0.826	0.877	0.894

Table 19: Results of the precision metric of the clustering algorithms for each threshold on the test split of the CORA data set.

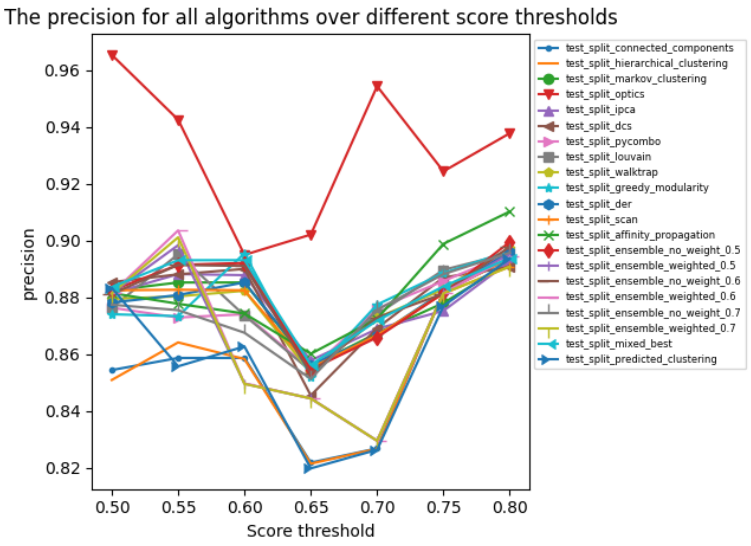

Figure 9: All algorithms on the CORA data set test split.

Algorithm	0.50	0.55	0.60	0.65	0.70	0.75	0.80
test_split_mixed_best	0.997	0.996	0.996	0.994	1.000	1.000	1.000
test_split_connected_components	1.000	1.000	1.000	1.000	1.000	1.000	1.000
test_split_hierarchical_clustering	0.907	0.967	0.987	0.989	0.998	1.000	1.000
test_split_markov_clustering	0.997	0.996	0.996	0.982	0.992	0.946	0.903
test_split_optics	0.402	0.291	0.330	0.202	0.255	0.208	0.288
test_split_ipca	0.979	0.943	0.925	0.908	0.816	0.717	0.643
test_split_dcs	0.976	0.984	0.986	0.886	0.849	0.883	0.815
test_split_combo	0.541	0.537	0.556	0.505	0.571	0.441	0.501
test_split_louvain	0.558	0.492	0.553	0.516	0.573	0.435	0.476
test_split_walktrap	0.942	0.884	0.882	0.883	0.750	0.671	0.597
test_split_greedy_modularity	0.523	0.539	0.529	0.508	0.577	0.511	0.508
test_split_der	0.603	0.612	0.627	0.697	0.578	0.506	0.548
test_split_scan	0.997	0.969	0.967	0.962	0.945	0.900	0.795
test_split_affinity_propagation	0.336	0.292	0.309	0.233	0.297	0.203	0.290
test_split_ensemble_no_weight_0.5	0.997	0.986	0.994	0.949	0.972	0.917	0.871
test_split_ensemble_weighted_0.5	0.762	0.491	0.843	0.939	0.966	0.967	0.854
test_split_ensemble_no_weight_0.6	0.986	0.983	0.979	0.924	0.953	0.873	0.749
test_split_ensemble_weighted_0.6	0.762	0.422	0.843	0.939	0.966	0.967	0.739
test_split_ensemble_no_weight_0.7	0.816	0.750	0.742	0.843	0.698	0.506	0.542
test_split_ensemble_weighted_0.7	0.762	0.381	0.843	0.939	0.966	0.967	0.680
test_split_predicted_clustering	0.841	0.925	0.980	0.914	0.988	0.899	0.991

Table 20: Results of the recall metric of the clustering algorithms for each threshold on the test split of the CORA data set.

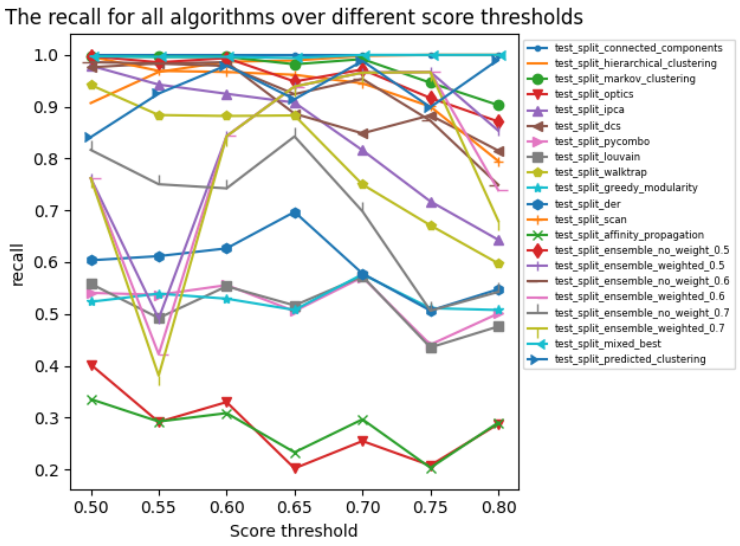

Figure 10: All algorithms on the CORA data set test split.

Algorithm	0.50	0.55	0.60	0.65	0.70	0.75	0.80
test_split_mixed_best	3.000	3.571	3.714	3.571	3.714	2.714	3.000
test_split_connected_components	3.429	4.429	4.429	4.143	4.571	3.429	3.429
test_split_hierarchical_clustering	10.429	9.143	6.429	4.857	4.857	3.429	3.429
test_split_markov_clustering	3.429	4.714	4.714	4.571	4.857	4.857	6.571
test_split_optics	126.286	163.429	128.143	102.714	62.571	52.286	55.429
test_split_ipca	4.714	7.571	10.286	12.000	12.714	14.000	14.714
test_split_dcs	3.714	5.143	5.857	5.857	8.714	5.857	7.571
test_split_combo	11.143	12.000	12.143	11.714	12.857	13.000	15.571
test_split_louvain	11.143	13.000	12.143	11.714	12.857	13.429	16.429
test_split_walktrap	3.714	6.857	8.714	7.857	11.429	12.143	20.143
test_split_greedy_modularity	11.429	12.000	12.000	11.857	12.429	12.143	14.714
test_split_der	29.429	30.286	29.714	30.000	31.571	31.143	33.714
test_split_scan	3.571	6.000	6.857	6.286	6.714	8.429	10.857
test_split_affinity_propagation	51.429	67.857	63.857	70.143	74.857	82.143	83.714
test_split_ensemble_no_weight_0.5	3.429	5.000	4.857	5.143	5.571	4.714	7.571
test_split_ensemble_weighted_0.5	15.714	60.571	26.429	7.857	8.571	6.714	21.286
test_split_ensemble_no_weight_0.6	3.857	5.429	5.571	6.286	8.000	6.286	10.143
test_split_ensemble_weighted_0.6	15.714	90.714	26.429	7.857	8.571	7.143	25.000
test_split_ensemble_no_weight_0.7	7.857	11.286	12.143	12.143	14.714	18.000	19.571
test_split_ensemble_weighted_0.7	15.714	108.714	26.714	7.857	8.571	7.429	28.143
test_split_predicted_clustering	14.000	7.429	8.714	10.143	5.286	8.571	4.571

Table 21: Results of the GMD metric of the clustering algorithms for each threshold on the test split of the CORA data set.

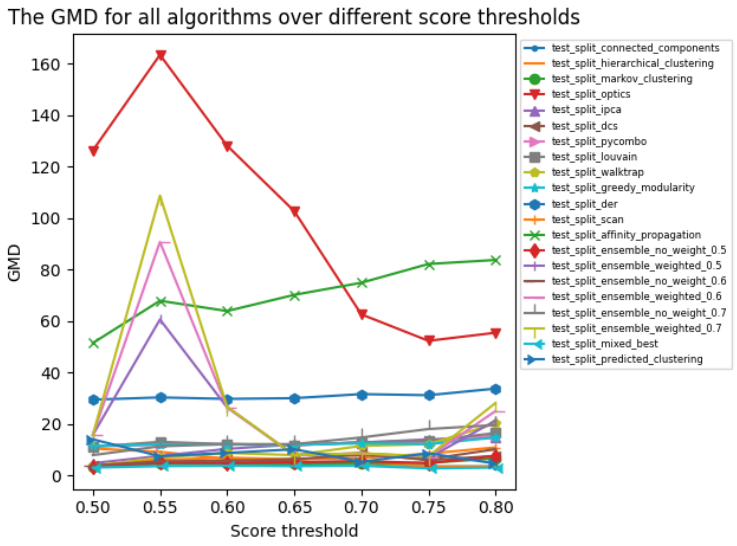

Figure 11: All algorithms on the CORA data set test split.

Algorithm	0.50	0.55	0.60	0.65	0.70	0.75	0.80
test_split_mixed_best	0.124	0.138	0.140	0.165	0.136	0.126	0.115
test_split_connected_components	0.158	0.187	0.187	0.219	0.173	0.136	0.118
test_split_hierarchical_clustering	0.285	0.238	0.214	0.234	0.176	0.136	0.118
test_split_markov_clustering	0.128	0.158	0.158	0.183	0.158	0.196	0.227
test_split_optics	1.500	1.822	1.607	1.665	1.249	1.379	1.254
test_split_ipca	0.152	0.216	0.250	0.300	0.345	0.453	0.520
test_split_dcs	0.148	0.170	0.182	0.275	0.288	0.230	0.307
test_split_combo	0.572	0.598	0.585	0.628	0.533	0.671	0.621
test_split_louvain	0.551	0.660	0.588	0.623	0.531	0.685	0.661
test_split_walktrap	0.168	0.263	0.285	0.299	0.397	0.498	0.592
test_split_greedy_modularity	0.588	0.597	0.602	0.626	0.516	0.574	0.603
test_split_der	0.674	0.659	0.644	0.637	0.687	0.732	0.693
test_split_scan	0.129	0.192	0.202	0.214	0.202	0.257	0.349
test_split_affinity_propagation	1.125	1.353	1.307	1.429	1.293	1.524	1.376
test_split_ensemble_no_weight_0.5	0.128	0.156	0.156	0.213	0.186	0.213	0.242
test_split_ensemble_weighted_0.5	0.423	0.929	0.442	0.256	0.231	0.183	0.346
test_split_ensemble_no_weight_0.6	0.143	0.162	0.172	0.250	0.221	0.253	0.361
test_split_ensemble_weighted_0.6	0.423	1.184	0.442	0.256	0.231	0.188	0.469
test_split_ensemble_no_weight_0.7	0.316	0.410	0.446	0.375	0.443	0.669	0.636
test_split_ensemble_weighted_0.7	0.423	1.373	0.444	0.256	0.231	0.191	0.564
test_split_predicted_clustering	0.355	0.273	0.246	0.339	0.200	0.259	0.141

Table 22: Results of the VoI metric of the clustering algorithms for each threshold on the test split of the CORA data set.

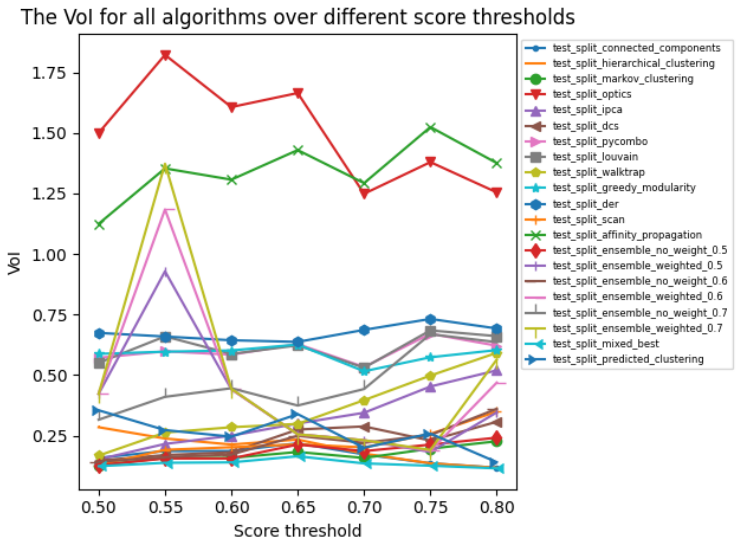

Figure 12: All algorithms on the CORA data set test split.

A.2 Geographical Settlements

Algorithm	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
mixed_best	0.703	0.657	0.642	0.586	0.557	0.541	0.479	0.422	0.397	0.334	0.279	0.247	0.207	0.171
connected_components	0.531	0.545	0.571	0.545	0.537	0.530	0.475	0.419	0.395	0.333	0.278	0.246	0.207	0.171
hierarchical_clustering	0.364	0.372	0.382	0.389	0.392	0.409	0.418	0.409	0.395	0.333	0.278	0.246	0.207	0.171
markov_clustering	0.644	0.608	0.601	0.554	0.539	0.524	0.469	0.418	0.391	0.333	0.278	0.246	0.207	0.171
optics	0.621	0.601	0.593	0.538	0.527	0.511	0.461	0.414	0.389	0.332	0.278	0.246	0.207	0.171
cdlib_ipca	0.535	0.531	0.530	0.521	0.511	0.489	0.429	0.390	0.372	0.317	0.272	0.244	0.201	0.171
cdlib_dcs	0.641	0.620	0.611	0.566	0.544	0.530	0.473	0.419	0.395	0.333	0.278	0.246	0.207	0.171
cdlib_combo	0.655	0.618	0.603	0.544	0.520	0.513	0.457	0.407	0.389	0.334	0.278	0.246	0.207	0.171
louvain	0.654	0.614	0.602	0.543	0.520	0.513	0.457	0.407	0.389	0.334	0.278	0.246	0.207	0.171
walktrap	0.646	0.619	0.611	0.555	0.540	0.521	0.468	0.418	0.391	0.332	0.278	0.246	0.207	0.171
greedy_modularity	0.654	0.615	0.602	0.543	0.520	0.513	0.457	0.407	0.389	0.334	0.278	0.246	0.207	0.171
cdlib_der	0.327	0.295	0.274	0.240	0.199	0.191	0.148	0.103	0.088	0.058	0.032	0.017	0.007	0.001
cdlib_scan	0.615	0.599	0.602	0.568	0.543	0.531	0.475	0.419	0.395	0.333	0.278	0.246	0.207	0.171
affinity_propagation	0.398	0.356	0.345	0.290	0.259	0.246	0.190	0.141	0.126	0.082	0.044	0.022	0.008	0.000
ensemble_no_weight_0.5	0.652	0.619	0.607	0.561	0.539	0.525	0.471	0.418	0.392	0.333	0.278	0.246	0.207	0.171
ensemble_no_weight_0.6	0.661	0.626	0.615	0.558	0.539	0.523	0.469	0.418	0.391	0.333	0.278	0.246	0.207	0.171
ensemble_no_weight_0.7	0.649	0.610	0.595	0.537	0.517	0.510	0.457	0.407	0.388	0.333	0.278	0.246	0.207	0.171

Table 23: Results of the F1 metric of the clustering algorithms for each threshold on the full data of the Geographical settlements data set.

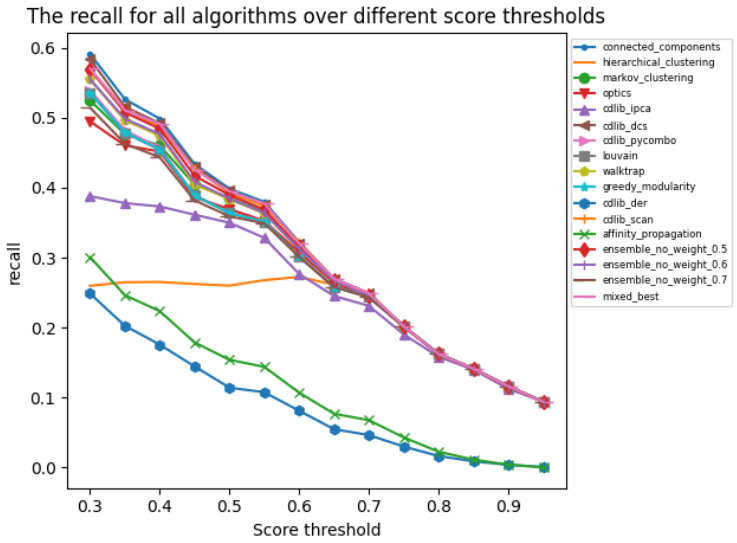

Figure 13: All algorithms on the Geographical settlements data set.

Algorithm	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
mixed_best	0.916	0.921	0.924	0.939	0.947	0.955	0.951	0.969	0.984	0.981	0.983	0.983	0.978	0.974
connected_components	0.482	0.564	0.668	0.734	0.820	0.879	0.900	0.941	0.959	0.954	0.961	0.958	0.950	0.941
hierarchical_clustering	0.605	0.624	0.682	0.750	0.798	0.867	0.899	0.940	0.959	0.954	0.961	0.958	0.950	0.941
markov_clustering	0.834	0.840	0.865	0.879	0.899	0.920	0.939	0.958	0.958	0.954	0.961	0.958	0.950	0.941
optics	0.832	0.866	0.863	0.884	0.922	0.930	0.925	0.954	0.963	0.958	0.961	0.958	0.950	0.941
cdlib_ipca	0.860	0.891	0.915	0.931	0.945	0.954	0.954	0.956	0.956	0.958	0.960	0.958	0.948	0.941
cdlib_dcs	0.710	0.776	0.804	0.828	0.869	0.897	0.910	0.941	0.959	0.954	0.961	0.958	0.950	0.941
cdlib_combo	0.835	0.858	0.887	0.896	0.914	0.945	0.947	0.956	0.966	0.963	0.961	0.958	0.950	0.941
louvain	0.839	0.857	0.889	0.897	0.914	0.945	0.947	0.956	0.966	0.963	0.961	0.958	0.950	0.941
walktrap	0.773	0.821	0.858	0.881	0.913	0.922	0.942	0.958	0.958	0.959	0.961	0.958	0.950	0.941
greedy_modularity	0.840	0.859	0.890	0.897	0.914	0.945	0.947	0.956	0.966	0.963	0.961	0.958	0.950	0.941
cdlib_der	0.478	0.546	0.626	0.706	0.765	0.850	0.868	0.917	0.920	0.898	0.912	0.884	0.748	0.339
cdlib_scan	0.667	0.721	0.785	0.843	0.882	0.914	0.906	0.941	0.959	0.954	0.961	0.958	0.950	0.941
affinity_propagation	0.589	0.641	0.758	0.770	0.824	0.865	0.874	0.909	0.944	0.938	0.984	1.000	1.000	0.000
ensemble_no_weight_0.5	0.763	0.792	0.812	0.856	0.875	0.913	0.936	0.954	0.958	0.954	0.961	0.958	0.950	0.941
ensemble_no_weight_0.6	0.816	0.839	0.864	0.880	0.902	0.925	0.942	0.958	0.958	0.954	0.961	0.958	0.950	0.941
ensemble_no_weight_0.7	0.877	0.893	0.905	0.906	0.921	0.947	0.947	0.956	0.958	0.963	0.961	0.958	0.950	0.941

Table 24: Results of the precision metric of the clustering algorithms for each threshold on the full data of the Geographical settlements data set.

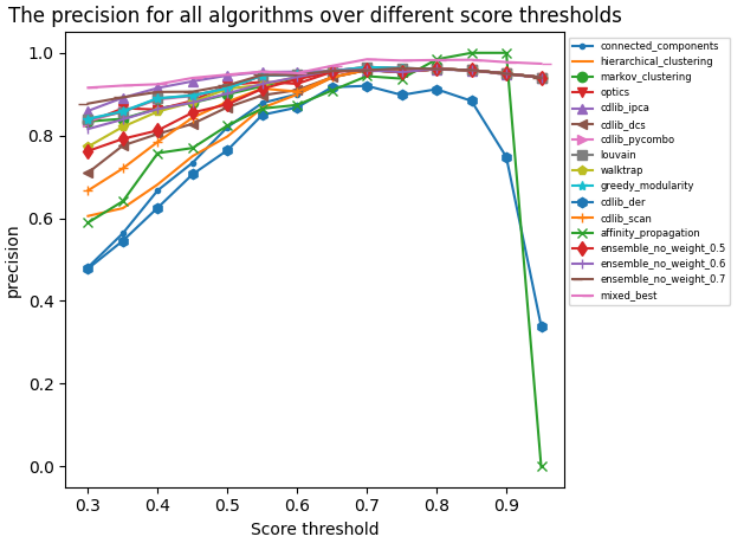

Figure 14: All algorithms on the Geographical settlements data set.

Algorithm	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
mixed_best	0.571	0.511	0.492	0.426	0.395	0.378	0.320	0.270	0.249	0.202	0.162	0.141	0.116	0.094
connected_components	0.592	0.527	0.499	0.434	0.399	0.380	0.322	0.270	0.249	0.202	0.162	0.141	0.116	0.094
hierarchical_clustering	0.260	0.265	0.265	0.262	0.260	0.268	0.273	0.261	0.249	0.202	0.162	0.141	0.116	0.094
markov_clustering	0.525	0.477	0.460	0.405	0.385	0.366	0.312	0.267	0.245	0.202	0.162	0.141	0.116	0.094
optics	0.495	0.461	0.452	0.387	0.369	0.353	0.307	0.264	0.244	0.201	0.162	0.141	0.116	0.094
cdlib_ipca	0.388	0.378	0.373	0.361	0.350	0.328	0.277	0.245	0.231	0.190	0.158	0.140	0.112	0.094
cdlib_dcs	0.584	0.516	0.492	0.430	0.396	0.376	0.320	0.270	0.249	0.202	0.162	0.141	0.116	0.094
cdlib_combo	0.539	0.482	0.457	0.390	0.364	0.352	0.302	0.258	0.244	0.202	0.162	0.141	0.116	0.094
louvain	0.536	0.479	0.454	0.389	0.364	0.352	0.302	0.258	0.244	0.202	0.162	0.141	0.116	0.094
walktrap	0.555	0.497	0.474	0.405	0.384	0.363	0.312	0.267	0.245	0.201	0.162	0.141	0.116	0.094
greedy_modularity	0.535	0.479	0.455	0.389	0.364	0.352	0.302	0.258	0.244	0.202	0.162	0.141	0.116	0.094
cdlib_der	0.249	0.203	0.176	0.144	0.114	0.107	0.081	0.055	0.046	0.030	0.016	0.009	0.004	0.001
cdlib_scan	0.570	0.512	0.488	0.428	0.392	0.374	0.322	0.270	0.249	0.202	0.162	0.141	0.116	0.094
affinity_propagation	0.301	0.246	0.224	0.179	0.154	0.144	0.107	0.077	0.067	0.043	0.022	0.011	0.004	0.000
ensemble_no_weight_0.5	0.569	0.508	0.485	0.417	0.389	0.369	0.315	0.268	0.246	0.202	0.162	0.141	0.116	0.094
ensemble_no_weight_0.6	0.555	0.499	0.477	0.409	0.385	0.364	0.312	0.267	0.245	0.202	0.162	0.141	0.116	0.094
ensemble_no_weight_0.7	0.515	0.463	0.443	0.382	0.359	0.349	0.301	0.258	0.243	0.201	0.162	0.141	0.116	0.094

Table 25: Results of the recall metric of the clustering algorithms for each threshold on the full data of the Geographical settlements data set.

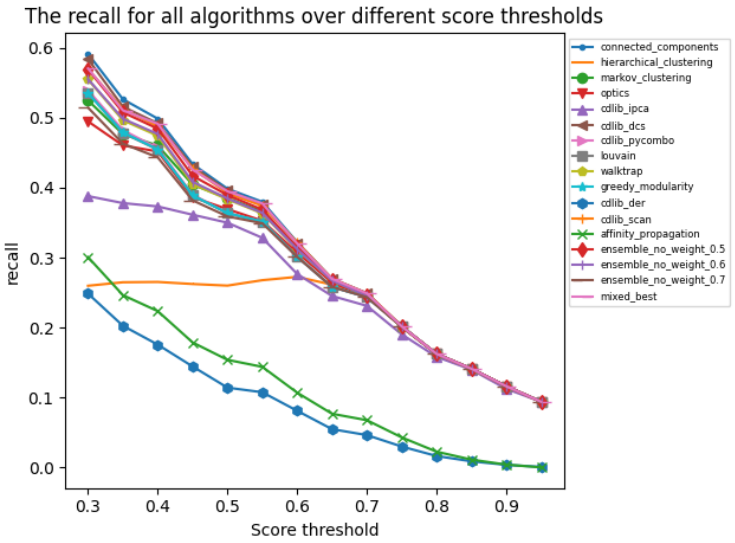

Figure 15: All algorithms on the Geographical settlements data set.

Algorithm	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
mixed_best	714.143	828.000	853.143	993.714	1043.143	1074.667	1203.714	1310.714	1361.143	1490.833	1598.000	1657.000	1744.286	1828.286
connected_components	827.000	914.000	920.000	1043.000	1075.000	1102.000	1221.000	1325.000	1373.000	1502.000	1610.000	1670.000	1757.000	1840.000
hierarchical_clustering	1421.000	1396.000	1391.000	1390.000	1374.000	1353.000	1340.000	1353.000	1373.000	1502.000	1610.000	1670.000	1757.000	1840.000
markov_clustering	861.000	946.000	956.000	1071.000	1084.000	1113.000	1230.000	1325.000	1377.000	1502.000	1610.000	1670.000	1757.000	1840.000
optics	1009.000	1042.000	1012.000	1138.000	1143.000	1157.000	1250.000	1340.000	1383.000	1505.000	1610.000	1670.000	1757.000	1840.000
cdlib_ipca	1098.000	1106.000	1091.000	1122.000	1124.000	1170.000	1286.000	1360.000	1403.000	1522.000	1619.000	1673.000	1765.000	1840.000
cdlib_dcs	811.000	920.000	928.000	1046.000	1079.000	1111.000	1225.000	1325.000	1373.000	1502.000	1610.000	1670.000	1757.000	1840.000
cdlib_combo	844.000	934.000	946.000	1085.000	1107.000	1128.000	1242.000	1335.000	1378.000	1501.000	1610.000	1670.000	1757.000	1840.000
louvain	847.000	941.000	950.000	1083.000	1107.000	1128.000	1242.000	1335.000	1378.000	1501.000	1610.000	1670.000	1757.000	1840.000
walktrap	851.000	939.000	938.000	1068.000	1088.000	1123.000	1233.000	1325.000	1377.000	1505.000	1610.000	1670.000	1757.000	1840.000
greedy_modularity	849.000	939.000	950.000	1083.000	1107.000	1128.000	1242.000	1335.000	1378.000	1501.000	1610.000	1670.000	1757.000	1840.000
cdlib_der	1499.714	1603.429	1644.714	1735.143	1807.857	1827.167	1926.714	2016.857	2046.000	2115.667	2169.714	2201.000	2223.571	2236.571
cdlib_scan	865.000	942.000	947.000	1053.000	1090.000	1116.000	1221.000	1325.000	1373.000	1502.000	1610.000	1670.000	1757.000	1840.000
affinity_propagation	1438.429	1565.000	1588.429	1702.571	1764.714	1778.333	1889.000	1982.571	2006.571	2088.000	2153.714	2193.571	2219.143	2234.000
ensemble_no_weight_0.5	824.000	920.286	928.714	1054.714	1083.714	1112.000	1227.714	1325.429	1375.857	1502.000	1610.000	1670.000	1757.000	1840.000
ensemble_no_weight_0.6	835.000	926.714	933.286	1063.000	1087.571	1117.333	1230.143	1325.000	1377.000	1502.000	1610.000	1670.000	1757.000	1840.000
ensemble_no_weight_0.7	877.571	964.000	974.714	1106.429	1119.857	1138.000	1243.429	1335.000	1379.571	1502.167	1610.000	1670.000	1757.000	1840.000

Table 26: Results of the GMD metric of the clustering algorithms for each threshold on the full data of the Geographical settlements data set.

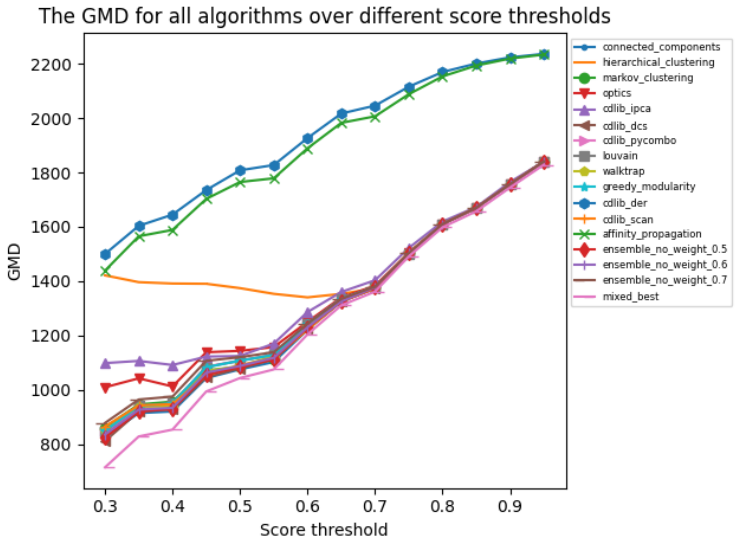

Figure 16: All algorithms on the Geographical settlements data set.

Algorithm	**0.30**	**0.35**	**0.40**	**0.45**	**0.50**	**0.55**	**0.60**	**0.65**	**0.70**	**0.75**	**0.80**	**0.85**	**0.90**	**0.95**
mixed_best	0.522	0.592	0.610	0.691	0.724	0.743	0.821	0.884	0.911	0.982	1.041	1.073	1.117	1.157
connected_components	0.706	0.716	0.694	0.751	0.759	0.768	0.835	0.893	0.918	0.989	1.047	1.080	1.123	1.163
hierarchical_clustering	0.990	0.967	0.955	0.947	0.934	0.914	0.902	0.907	0.918	0.989	1.047	1.080	1.123	1.163
markov_clustering	0.624	0.676	0.681	0.746	0.756	0.771	0.838	0.892	0.922	0.989	1.047	1.080	1.123	1.163
optics	0.692	0.711	0.704	0.778	0.780	0.792	0.850	0.900	0.924	0.991	1.047	1.080	1.123	1.163
cdlib_ipca	0.772	0.772	0.763	0.775	0.779	0.805	0.875	0.917	0.940	1.003	1.053	1.082	1.128	1.163
cdlib_dcs	0.621	0.671	0.673	0.737	0.755	0.770	0.837	0.893	0.918	0.989	1.047	1.080	1.123	1.163
cdlib_combo	0.613	0.665	0.675	0.756	0.774	0.780	0.847	0.901	0.922	0.988	1.047	1.080	1.123	1.163
louvain	0.614	0.670	0.677	0.755	0.774	0.780	0.847	0.901	0.922	0.988	1.047	1.080	1.123	1.163
walktrap	0.628	0.672	0.671	0.745	0.755	0.776	0.839	0.892	0.922	0.990	1.047	1.080	1.123	1.163
greedy_modularity	0.616	0.669	0.676	0.755	0.774	0.780	0.847	0.901	0.922	0.988	1.047	1.080	1.123	1.163
cdlib_der	1.060	1.084	1.092	1.123	1.156	1.158	1.202	1.242	1.256	1.287	1.312	1.326	1.336	1.342
cdlib_scan	0.662	0.692	0.685	0.736	0.758	0.769	0.834	0.893	0.918	0.989	1.047	1.080	1.123	1.163
affinity_propagation	0.986	1.036	1.033	1.088	1.113	1.119	1.173	1.217	1.229	1.269	1.301	1.321	1.334	1.341
ensemble_no_weight_0.5	0.614	0.667	0.673	0.739	0.758	0.771	0.836	0.893	0.921	0.989	1.047	1.080	1.123	1.163
ensemble_no_weight_0.6	0.609	0.660	0.666	0.741	0.757	0.773	0.838	0.892	0.922	0.989	1.047	1.080	1.123	1.163
ensemble_no_weight_0.7	0.625	0.676	0.686	0.764	0.779	0.784	0.847	0.901	0.924	0.989	1.047	1.080	1.123	1.163

Table 27: Results of the VoI metric of the clustering algorithms for each threshold on the full data of the Geographical settlements data set.

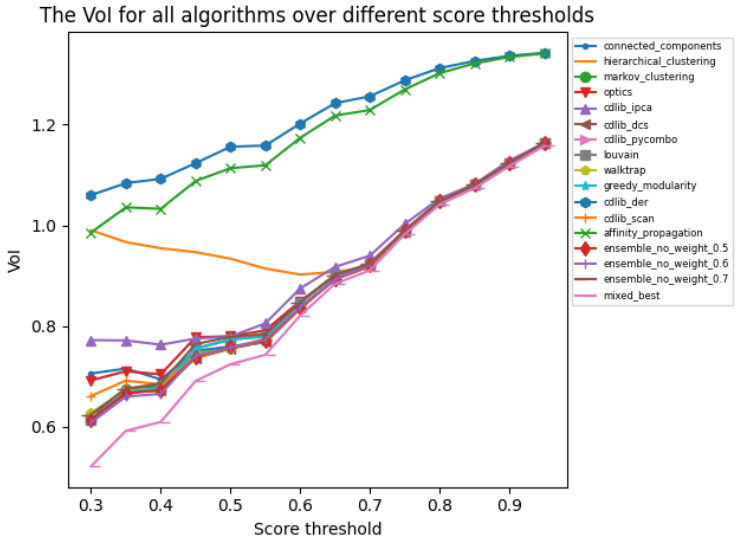

Figure 17: All algorithms on the Geographical settlements data set.

Geographical settlements - Test split

Algorithm	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
test_split_mixed_best	0.933	0.955	0.952	0.962	0.965	0.981	0.970	0.992	0.992	0.995	0.986	0.999	0.980	0.989
test_split_connected_components	0.670	0.777	0.833	0.841	0.891	0.952	0.941	0.978	0.983	0.979	0.977	0.990	0.965	0.965
test_split_hierarchical_clustering	0.518	0.594	0.618	0.664	0.715	0.785	0.870	0.965	0.983	0.979	0.977	0.990	0.965	0.965
test_split_markov_clustering	0.858	0.883	0.896	0.900	0.923	0.956	0.951	0.982	0.976	0.979	0.977	0.990	0.965	0.965
test_split_optics	0.826	0.876	0.885	0.877	0.915	0.944	0.935	0.973	0.976	0.979	0.977	0.990	0.965	0.965
test_split_ipca	0.737	0.808	0.815	0.877	0.911	0.916	0.905	0.937	0.951	0.955	0.967	0.983	0.947	0.965
test_split_dcs	0.825	0.871	0.899	0.899	0.916	0.955	0.942	0.978	0.983	0.979	0.977	0.990	0.965	0.965
test_split_combo	0.868	0.898	0.902	0.893	0.906	0.944	0.942	0.963	0.974	0.986	0.977	0.990	0.965	0.965
test_split_louvain	0.869	0.889	0.903	0.892	0.906	0.944	0.942	0.963	0.974	0.986	0.977	0.990	0.965	0.965
test_split_walktrap	0.842	0.892	0.907	0.898	0.931	0.957	0.952	0.982	0.976	0.980	0.977	0.990	0.965	0.965
test_split_greedy_modularity	0.869	0.892	0.903	0.892	0.906	0.944	0.942	0.963	0.974	0.986	0.977	0.990	0.965	0.965
test_split_der	0.460	0.471	0.459	0.444	0.403	0.438	0.395	0.328	0.300	0.275	0.168	0.097	0.058	0.030
test_split_scan	0.794	0.836	0.883	0.904	0.922	0.960	0.944	0.978	0.983	0.979	0.977	0.990	0.965	0.965
test_split_affinity_propagation	0.555	0.548	0.561	0.529	0.525	0.548	0.494	0.443	0.416	0.367	0.209	0.129	0.056	0.000
test_split_ensemble_no_weight_0.5	0.851	0.884	0.885	0.898	0.916	0.957	0.954	0.979	0.976	0.979	0.977	0.990	0.965	0.965
test_split_ensemble_weighted_0.5	0.821	0.888	0.874	0.863	0.898	0.953	0.944	0.455	0.490	0.368	0.168	0.097	0.442	0.030
test_split_ensemble_no_weight_0.6	0.867	0.901	0.914	0.903	0.920	0.957	0.952	0.982	0.976	0.979	0.977	0.990	0.965	0.965
test_split_ensemble_weighted_0.6	0.847	0.885	0.878	0.893	0.901	0.952	0.938	0.455	0.490	0.368	0.168	0.097	0.429	0.030
test_split_ensemble_no_weight_0.7	0.871	0.891	0.899	0.887	0.902	0.942	0.942	0.963	0.974	0.985	0.977	0.990	0.965	0.965
test_split_ensemble_weighted_0.7	0.861	0.889	0.889	0.906	0.898	0.909	0.927	0.455	0.489	0.368	0.166	0.097	0.429	0.030
test_split_predicted_clustering	0.812	0.844	0.849	0.865	0.832	0.843	0.847	0.847	0.825	0.836	0.852	0.468	0.901	0.701

Table 28: Results of the F1 metric of the clustering algorithms for each threshold on the test split of the Geographical settlements data set.

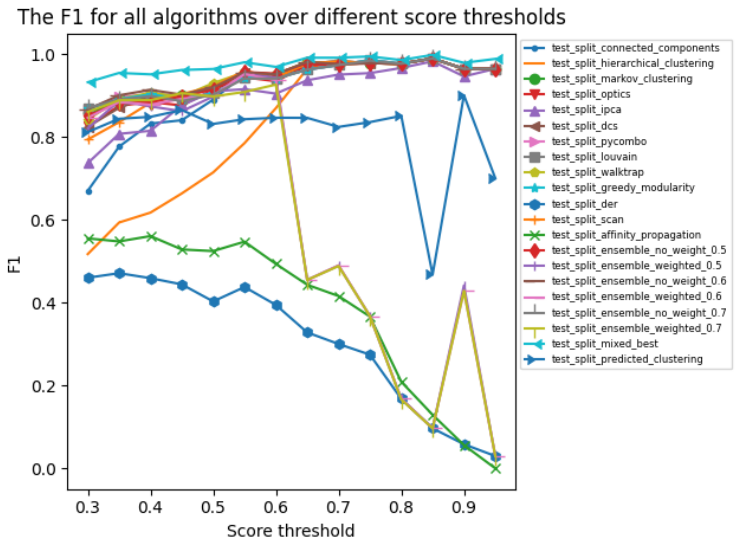

Figure 18: All algorithms on the Geographical settlements data set test split.

Algorithm	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
test_split_mixed_best	0.901	0.934	0.918	0.944	0.940	0.968	0.948	0.984	0.986	0.991	0.973	0.998	0.960	0.978
test_split_connected_components	0.512	0.648	0.724	0.734	0.809	0.910	0.889	0.957	0.967	0.959	0.956	0.981	0.933	0.933
test_split_hierarchical_clustering	0.655	0.736	0.749	0.754	0.799	0.905	0.891	0.958	0.967	0.959	0.956	0.981	0.933	0.933
test_split_markov_clustering	0.835	0.854	0.869	0.881	0.881	0.948	0.932	0.973	0.967	0.959	0.956	0.981	0.933	0.933
test_split_optics	0.831	0.865	0.847	0.879	0.905	0.943	0.918	0.967	0.967	0.965	0.956	0.981	0.933	0.933
test_split_ipca	0.863	0.902	0.899	0.944	0.936	0.968	0.948	0.972	0.965	0.966	0.955	0.980	0.931	0.933
test_split_dcs	0.712	0.785	0.826	0.823	0.850	0.924	0.896	0.957	0.967	0.959	0.956	0.981	0.933	0.933
test_split_combo	0.831	0.880	0.885	0.902	0.901	0.962	0.940	0.972	0.967	0.973	0.956	0.981	0.933	0.933
test_split_louvain	0.839	0.873	0.887	0.901	0.901	0.962	0.940	0.972	0.967	0.973	0.956	0.981	0.933	0.933
test_split_walktrap	0.774	0.843	0.863	0.876	0.897	0.953	0.936	0.973	0.967	0.967	0.956	0.981	0.933	0.933
test_split_greedy_modularity	0.840	0.876	0.888	0.901	0.901	0.962	0.940	0.972	0.967	0.973	0.956	0.981	0.933	0.933
test_split_der	0.518	0.659	0.718	0.709	0.742	0.904	0.855	0.929	0.939	0.919	0.878	0.976	0.676	0.536
test_split_scan	0.679	0.737	0.810	0.840	0.870	0.933	0.895	0.957	0.967	0.959	0.956	0.981	0.933	0.933
test_split_affinity_propagation	0.614	0.676	0.793	0.763	0.798	0.913	0.878	0.949	0.953	0.959	0.960	1.000	1.000	0.000
test_split_ensemble_no_weight_0.5	0.774	0.816	0.812	0.853	0.857	0.947	0.931	0.967	0.967	0.959	0.956	0.981	0.933	0.933
test_split_ensemble_weighted_0.5	0.715	0.872	0.802	0.765	0.840	0.956	0.932	0.947	0.951	0.915	0.886	0.976	0.867	0.536
test_split_ensemble_no_weight_0.6	0.814	0.860	0.870	0.880	0.876	0.951	0.934	0.973	0.967	0.959	0.956	0.981	0.933	0.933
test_split_ensemble_weighted_0.6	0.770	0.883	0.815	0.815	0.846	0.956	0.934	0.947	0.951	0.915	0.886	0.976	0.843	0.536
test_split_ensemble_no_weight_0.7	0.877	0.900	0.900	0.912	0.907	0.963	0.940	0.972	0.967	0.973	0.956	0.981	0.933	0.933
test_split_ensemble_weighted_0.7	0.818	0.895	0.849	0.842	0.847	0.957	0.945	0.947	0.951	0.915	0.885	0.976	0.843	0.536
test_split_predicted_clustering	0.810	0.848	0.863	0.894	0.878	0.946	0.916	0.970	0.963	0.969	0.979	0.986	0.946	0.843

Table 29: Results of the precision metric of the clustering algorithms for each threshold on the test split of the Geographical settlements data set.

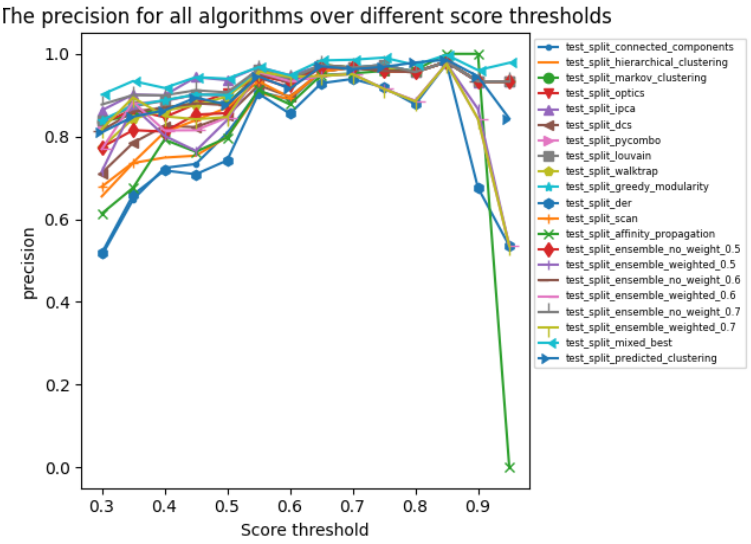

Figure 19: All algorithms on the Geographical settlements data set test split.

Algorithm	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
test_split_mixed_best	0.967	0.978	0.989	0.982	0.991	0.994	0.993	1.000	0.998	0.999	1.000	1.000	1.000	1.000
test_split_connected_components	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
test_split_hierarchical_clustering	0.435	0.508	0.534	0.601	0.656	0.695	0.851	0.973	1.000	1.000	1.000	1.000	1.000	1.000
test_split_markov_clustering	0.885	0.918	0.926	0.921	0.969	0.964	0.971	0.990	0.987	1.000	1.000	1.000	1.000	1.000
test_split_optics	0.824	0.887	0.928	0.875	0.925	0.947	0.953	0.980	0.987	0.993	1.000	1.000	1.000	1.000
test_split_ipca	0.646	0.732	0.747	0.820	0.887	0.869	0.867	0.907	0.938	0.944	0.980	0.986	0.964	1.000
test_split_dcs	0.984	0.981	0.989	0.993	0.993	0.989	0.995	1.000	1.000	1.000	1.000	1.000	1.000	1.000
test_split_combo	0.910	0.917	0.922	0.886	0.912	0.928	0.944	0.955	0.982	1.000	1.000	1.000	1.000	1.000
test_split_louvain	0.904	0.908	0.921	0.883	0.912	0.928	0.944	0.955	0.982	1.000	1.000	1.000	1.000	1.000
test_split_walktrap	0.928	0.948	0.959	0.921	0.967	0.960	0.968	0.990	0.987	0.993	1.000	1.000	1.000	1.000
test_split_greedy_modularity	0.902	0.910	0.921	0.883	0.912	0.928	0.944	0.955	0.982	1.000	1.000	1.000	1.000	1.000
test_split_der	0.426	0.381	0.347	0.331	0.281	0.289	0.258	0.199	0.179	0.162	0.093	0.051	0.031	0.015
test_split_scan	0.963	0.976	0.975	0.981	0.982	0.989	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
test_split_affinity_propagation	0.516	0.469	0.440	0.410	0.393	0.396	0.345	0.291	0.270	0.230	0.118	0.070	0.029	0.000
test_split_ensemble_no_weight_0.5	0.950	0.968	0.979	0.950	0.985	0.968	0.977	0.993	0.987	1.000	1.000	1.000	1.000	1.000
test_split_ensemble_weighted_0.5	0.969	0.906	0.970	0.997	0.968	0.952	0.958	0.331	0.365	0.277	0.093	0.051	0.434	0.015
test_split_ensemble_no_weight_0.6	0.929	0.948	0.965	0.928	0.970	0.962	0.971	0.990	0.987	1.000	1.000	1.000	1.000	1.000
test_split_ensemble_weighted_0.6	0.948	0.889	0.960	0.989	0.965	0.950	0.944	0.331	0.365	0.277	0.093	0.051	0.427	0.015
test_split_ensemble_no_weight_0.7	0.866	0.882	0.899	0.864	0.899	0.922	0.944	0.955	0.982	0.997	1.000	1.000	1.000	1.000
test_split_ensemble_weighted_0.7	0.911	0.885	0.938	0.983	0.958	0.878	0.912	0.331	0.364	0.277	0.092	0.051	0.427	0.015
test_split_predicted_clustering	0.824	0.847	0.839	0.840	0.793	0.764	0.789	0.752	0.731	0.746	0.781	0.307	0.862	0.711

Table 30: Results of the recall metric of the clustering algorithms for each threshold on the test split of the Geographical settlements data set.

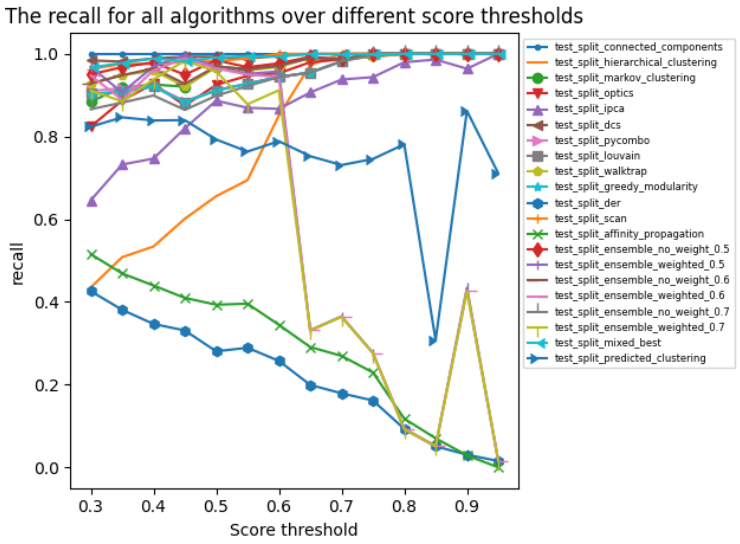

Figure 20: All algorithms on the Geographical settlements data set test split.

Algorithm	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
test_split_mixed_best	18.429	12.000	9.143	7.429	6.000	3.667	4.143	1.429	2.143	1.167	1.714	0.143	1.857	1.000
test_split_connected_components	39.714	30.143	23.286	19.000	14.000	7.833	9.143	4.286	3.571	3.000	3.857	2.286	4.429	4.000
test_split_hierarchical_clustering	163.286	127.286	114.286	90.143	72.000	60.833	31.286	8.714	3.571	3.000	3.857	2.286	4.429	4.000
test_split_markov_clustering	47.571	33.714	30.000	25.857	15.571	10.167	10.571	4.000	4.286	3.000	3.857	2.286	4.429	4.000
test_split_optics	79.286	53.429	38.000	41.143	29.143	15.167	16.000	7.571	4.286	4.000	3.857	2.286	4.429	4.000
test_split_ipca	100.143	65.143	60.429	35.286	21.429	21.833	21.714	11.143	9.000	7.333	5.286	3.143	6.286	4.000
test_split_dcs	36.857	31.143	24.000	19.286	15.000	9.833	9.714	4.286	3.571	3.000	3.857	2.286	4.429	4.000
test_split_combo	44.286	33.143	28.143	27.714	20.429	13.333	12.714	6.143	4.571	2.667	3.857	2.286	4.429	4.000
test_split_louvain	45.000	35.000	28.286	27.429	20.429	13.333	12.714	6.143	4.571	2.667	3.857	2.286	4.429	4.000
test_split_walktrap	46.429	34.286	26.143	25.571	16.143	11.167	11.429	4.000	4.286	4.000	3.857	2.286	4.429	4.000
test_split_greedy_modularity	45.429	34.429	28.286	27.429	20.429	13.333	12.714	6.143	4.571	2.667	3.857	2.286	4.429	4.000
test_split_der	176.714	170.143	164.571	159.857	163.714	153.500	148.714	145.000	139.714	126.333	116.857	110.000	98.286	82.000
test_split_scan	45.429	35.714	28.714	22.714	17.571	10.000	8.857	4.286	3.571	3.000	3.857	2.286	4.429	4.000
test_split_affinity_propagation	163.429	162.714	153.714	154.714	150.714	141.500	139.571	135.857	130.429	120.833	114.857	109.143	96.857	82.286
test_split_ensemble_no_weight_0.5	40.714	30.143	23.857	22.429	15.143	10.000	10.286	4.143	4.286	3.000	3.857	2.286	4.429	4.000
test_split_ensemble_weighted_0.5	39.000	33.714	25.429	19.143	17.286	12.500	13.429	123.143	113.714	111.167	117.143	110.143	58.286	82.000
test_split_ensemble_no_weight_0.6	44.286	31.571	24.857	24.429	16.571	10.333	10.571	4.000	4.286	3.000	3.857	2.286	4.429	4.000
test_split_ensemble_weighted_0.6	39.714	35.429	25.857	19.714	17.571	13.000	14.714	123.143	113.714	111.167	117.143	110.143	59.000	82.000
test_split_ensemble_no_weight_0.7	51.571	39.571	32.857	33.000	23.571	15.333	12.857	6.143	4.571	3.000	3.857	2.286	4.429	4.000
test_split_ensemble_weighted_0.7	43.429	35.429	28.571	21.000	18.571	30.333	19.143	123.143	114.000	111.167	117.286	110.143	59.000	82.000
test_split_predicted_clustering	74.714	63.143	58.429	45.286	55.857	50.333	38.571	32.857	37.857	31.500	24.286	81.571	17.571	26.429

Table 31: Results of the GMD metric of the clustering algorithms for each threshold on the test split of the Geographical settlements data set.

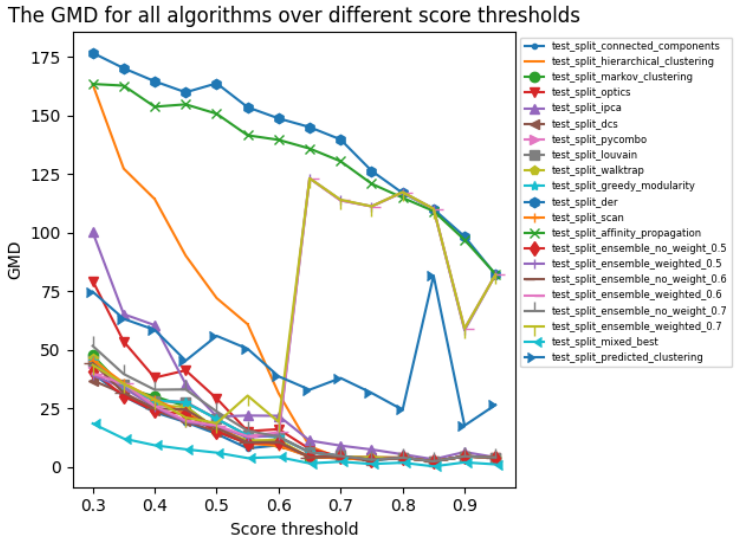

Figure 21: All algorithms on the Geographical settlements data set test split.

Algorithm	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
test_split_mixed_best	0.074	0.046	0.040	0.030	0.025	0.014	0.017	0.005	0.006	0.003	0.006	0.000	0.006	0.003
test_split_connected_components	0.256	0.167	0.118	0.100	0.068	0.032	0.036	0.014	0.011	0.010	0.011	0.005	0.012	0.011
test_split_hierarchical_clustering	0.542	0.419	0.373	0.296	0.234	0.189	0.099	0.026	0.011	0.010	0.011	0.005	0.012	0.011
test_split_markov_clustering	0.174	0.126	0.109	0.095	0.061	0.037	0.037	0.013	0.014	0.010	0.011	0.005	0.012	0.011
test_split_optics	0.247	0.164	0.127	0.131	0.088	0.051	0.051	0.021	0.014	0.012	0.011	0.005	0.012	0.011
test_split_ipca	0.332	0.220	0.203	0.120	0.077	0.075	0.072	0.039	0.030	0.024	0.015	0.008	0.018	0.011
test_split_dcs	0.169	0.130	0.096	0.082	0.062	0.036	0.037	0.014	0.011	0.010	0.011	0.005	0.012	0.011
test_split_combo	0.165	0.119	0.104	0.101	0.079	0.048	0.044	0.022	0.015	0.008	0.011	0.005	0.012	0.011
test_split_louvain	0.165	0.127	0.104	0.102	0.079	0.048	0.044	0.022	0.015	0.008	0.011	0.005	0.012	0.011
test_split_walktrap	0.182	0.124	0.097	0.095	0.059	0.039	0.038	0.013	0.014	0.012	0.011	0.005	0.012	0.011
test_split_greedy_modularity	0.167	0.125	0.103	0.102	0.079	0.048	0.044	0.022	0.015	0.008	0.011	0.005	0.012	0.011
test_split_der	0.608	0.538	0.509	0.474	0.465	0.430	0.397	0.370	0.352	0.311	0.276	0.255	0.225	0.186
test_split_scan	0.206	0.151	0.112	0.085	0.065	0.034	0.035	0.014	0.011	0.010	0.011	0.005	0.012	0.011
test_split_affinity_propagation	0.534	0.495	0.454	0.440	0.413	0.383	0.363	0.338	0.322	0.292	0.269	0.252	0.221	0.187
test_split_ensemble_no_weight_0.5	0.166	0.120	0.101	0.088	0.063	0.036	0.035	0.014	0.014	0.010	0.011	0.005	0.012	0.011
test_split_ensemble_weighted_0.5	0.177	0.124	0.109	0.093	0.072	0.042	0.044	0.312	0.282	0.273	0.276	0.255	0.135	0.186
test_split_ensemble_no_weight_0.6	0.165	0.114	0.091	0.091	0.065	0.037	0.036	0.013	0.014	0.010	0.011	0.005	0.012	0.011
test_split_ensemble_weighted_0.6	0.165	0.128	0.108	0.084	0.072	0.044	0.048	0.312	0.282	0.273	0.276	0.255	0.136	0.186
test_split_ensemble_no_weight_0.7	0.176	0.133	0.113	0.112	0.085	0.053	0.044	0.022	0.015	0.009	0.011	0.005	0.012	0.011
test_split_ensemble_weighted_0.7	0.166	0.127	0.111	0.081	0.074	0.087	0.059	0.312	0.282	0.273	0.277	0.255	0.136	0.186
test_split_predicted_clustering	0.245	0.197	0.180	0.142	0.165	0.151	0.120	0.101	0.108	0.087	0.063	0.189	0.041	0.061

Table 32: Results of the VoI metric of the clustering algorithms for each threshold on the test split of the Geographical settlements data set.

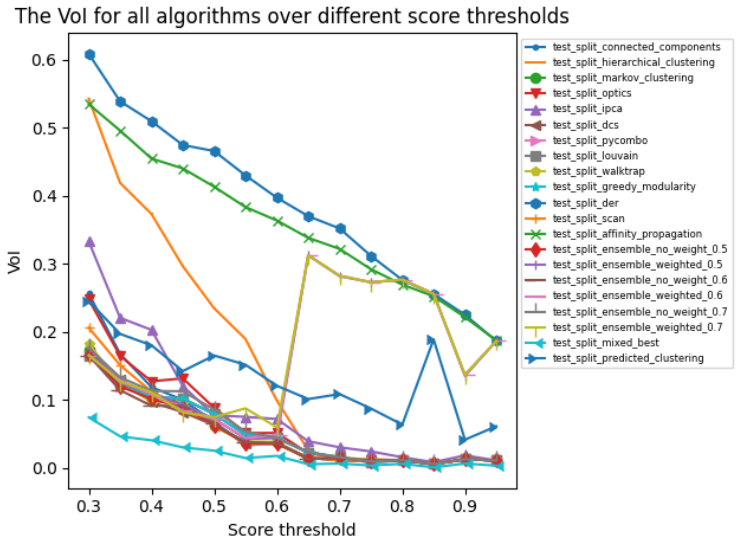

Figure 22: All algorithms on the Geographical settlements data set test split.

A.3 Musicbrainz

Algorithm	0.3	0.35	0.4	0.45	0.5	0.55
mixed_best	0.693	0.688	0.682	0.673	0.665	0.654
connected_components	0.504	0.580	0.592	0.617	0.627	0.629
hierarchical_clustering	0.514	0.526	0.525	0.532	0.546	0.571
markov_clustering	0.657	0.661	0.660	0.656	0.652	0.642
optics	0.653	0.655	0.653	0.644	0.635	0.621
cdlib_ipca	0.629	0.629	0.621	0.610	0.595	0.579
cdlib_dcs	0.590	0.627	0.651	0.644	0.644	0.638
cdlib_combo	0.651	0.661	0.658	0.656	0.650	0.641
louvain	0.651	0.660	0.659	0.656	0.650	0.641
walktrap	0.638	0.658	0.653	0.652	0.649	0.641
greedy_modularity	0.652	0.662	0.659	0.655	0.651	0.641
cdlib_der	0.232	0.247	0.244	0.248	0.249	0.244
cdlib_scan	0.607	0.629	0.634	0.643	0.644	0.640
affinity_propagation	0.267	0.275	0.273	0.276	0.274	0.265
ensemble_no_weight_0.5	0.607	0.652	0.651	0.652	0.646	0.640
ensemble_no_weight_0.6	0.647	0.656	0.660	0.654	0.648	0.641
ensemble_no_weight_0.7	0.666	0.669	0.664	0.655	0.651	0.640

Table 33: Results of the F1 metric of the clustering algorithms for each threshold on the full data of the Musicbrainz data set.

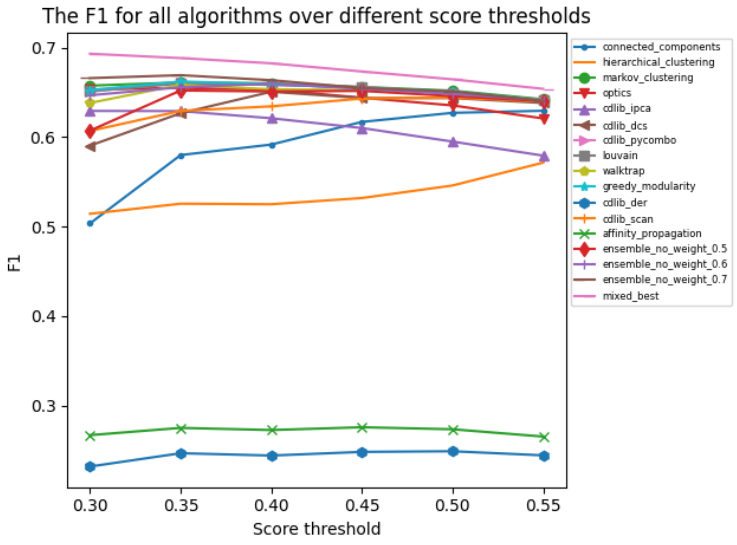

Figure 23: All algorithms on the Musicbrainz data set.

Algorithm	0.3	0.35	0.4	0.45	0.5	0.55
mixed_best	0.957	0.958	0.960	0.963	0.961	0.964
connected_components	0.466	0.626	0.666	0.758	0.814	0.862
hierarchical_clustering	0.828	0.858	0.819	0.805	0.819	0.860
markov_clustering	0.845	0.871	0.881	0.895	0.911	0.919
optics	0.878	0.895	0.912	0.921	0.933	0.936
cdlib_ipca	0.905	0.931	0.943	0.949	0.953	0.959
cdlib_dcs	0.645	0.750	0.841	0.847	0.877	0.901
cdlib_combo	0.819	0.875	0.889	0.917	0.928	0.945
louvain	0.821	0.869	0.893	0.918	0.928	0.945
walktrap	0.776	0.859	0.861	0.891	0.920	0.932
greedy_modularity	0.824	0.875	0.893	0.913	0.931	0.943
cdlib_der	0.364	0.522	0.554	0.676	0.760	0.822
cdlib_scan	0.687	0.762	0.792	0.852	0.884	0.911
affinity_propagation	0.450	0.574	0.629	0.758	0.813	0.852
ensemble_no_weight_0.5	0.685	0.828	0.842	0.877	0.887	0.909
ensemble_no_weight_0.6	0.798	0.844	0.878	0.891	0.905	0.922
ensemble_no_weight_0.7	0.878	0.912	0.918	0.919	0.941	0.949

Table 34: Results of the precision metric of the clustering algorithms for each threshold on the full data of the Musicbrainz data set.

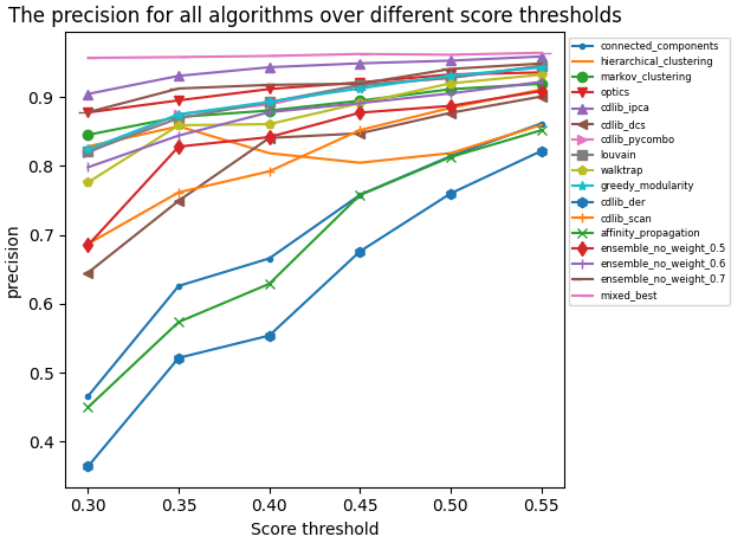

Figure 24: All algorithms on the Musicbrainz data set.

Algorithm	0.3	0.35	0.4	0.45	0.5	0.55
mixed_best	0.543	0.537	0.529	0.518	0.508	0.495
connected_components	0.547	0.540	0.532	0.520	0.510	0.496
hierarchical_clustering	0.373	0.379	0.386	0.397	0.409	0.428
markov_clustering	0.538	0.533	0.528	0.518	0.507	0.493
optics	0.520	0.516	0.509	0.495	0.482	0.464
cdlib_ipca	0.482	0.475	0.463	0.450	0.432	0.415
cdlib_dcs	0.544	0.538	0.531	0.519	0.508	0.494
cdlib_combo	0.540	0.532	0.522	0.511	0.500	0.485
louvain	0.540	0.532	0.522	0.510	0.500	0.485
walktrap	0.542	0.534	0.526	0.515	0.502	0.489
greedy_modularity	0.540	0.532	0.522	0.510	0.501	0.485
cdlib_der	0.170	0.162	0.157	0.152	0.149	0.144
cdlib_scan	0.543	0.536	0.529	0.516	0.507	0.493
affinity_propagation	0.190	0.181	0.174	0.169	0.164	0.157
ensemble_no_weight_0.5	0.545	0.538	0.531	0.518	0.508	0.494
ensemble_no_weight_0.6	0.544	0.536	0.528	0.516	0.505	0.492
ensemble_no_weight_0.7	0.536	0.528	0.520	0.508	0.498	0.483

Table 35: Results of the recall metric of the clustering algorithms for each threshold on the full data of the Musicbrainz data set.

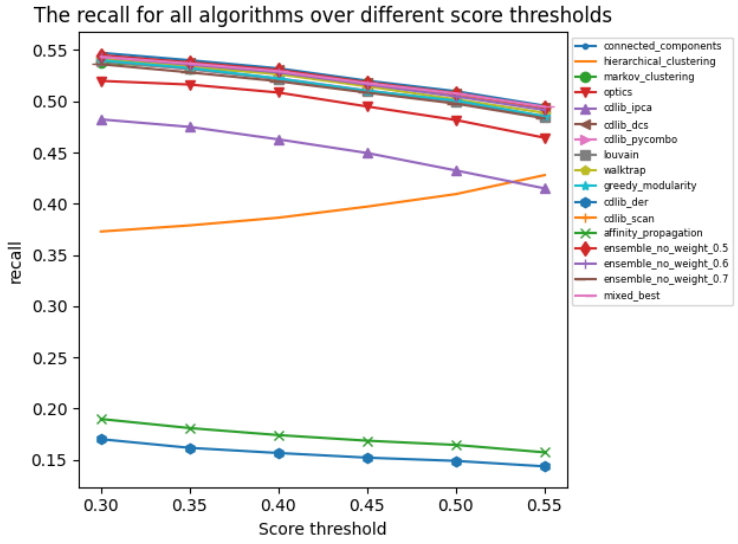

Figure 25: All algorithms on the Musicbrainz data set.

Algorithm	0.3	0.35	0.4	0.45	0.5	0.55
mixed_best	3596.143	3656.143	3720.286	3816.857	3918.429	4032.714
connected_components	4017.000	3949.000	3926.000	3993.000	4053.000	4152.000
hierarchical_clustering	5165.000	5096.000	5036.000	4978.000	4898.000	4774.000
markov_clustering	3885.000	3886.000	3888.000	3963.000	4042.000	4145.000
optics	4142.000	4089.000	4077.000	4164.000	4237.000	4369.000
cdlib_ipca	4198.000	4203.000	4247.000	4347.000	4489.000	4625.000
cdlib_dcs	4031.000	3957.000	3932.000	4004.000	4065.000	4165.000
cdlib_combo	3964.000	3933.000	3932.000	4008.000	4080.000	4195.000
louvain	3963.000	3939.000	3942.000	4015.000	4081.000	4196.000
walktrap	3968.000	3929.000	3916.000	3997.000	4076.000	4180.000
greedy_modularity	3954.000	3924.000	3930.000	4010.000	4069.000	4190.000
cdlib_der	7398.143	7369.714	7363.571	7397.571	7408.857	7467.571
cdlib_scan	4013.000	3966.143	3943.000	4015.000	4068.000	4161.000
affinity_propagation	7564.286	7545.714	7543.000	7562.286	7564.714	7609.000
ensemble_no_weight_0.5	3954.571	3915.571	3906.143	3977.000	4050.143	4148.286
ensemble_no_weight_0.6	3937.429	3914.571	3902.857	3976.286	4058.143	4158.286
ensemble_no_weight_0.7	3916.714	3917.714	3927.429	4015.429	4086.429	4199.000

Table 36: Results of the GMD metric of the clustering algorithms for each threshold on the full data of the Musicbrainz data set.

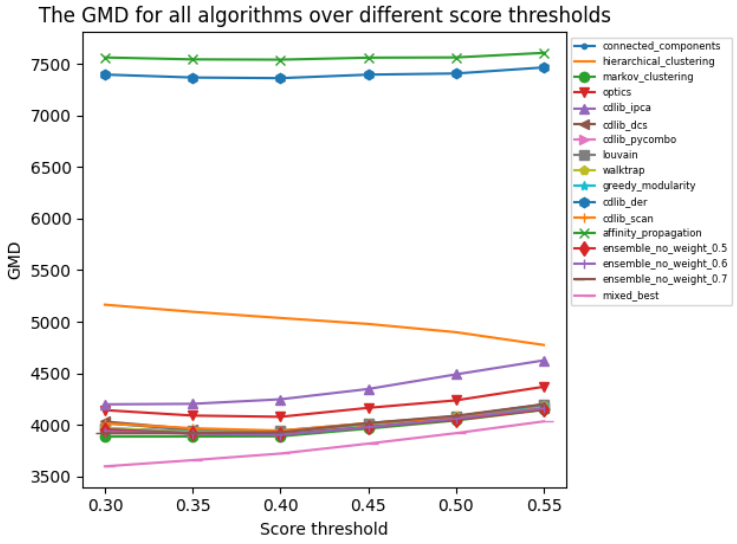

Figure 26: All algorithms on the Musicbrainz data set.

Algorithm	0.3	0.35	0.4	0.45	0.5	0.55
mixed_best	0.350	0.355	0.360	0.368	0.377	0.387
connected_components	0.437	0.410	0.402	0.402	0.402	0.407
hierarchical_clustering	0.502	0.494	0.488	0.484	0.475	0.460
markov_clustering	0.385	0.383	0.382	0.388	0.392	0.401
optics	0.402	0.396	0.395	0.402	0.408	0.420
cdlib_ipca	0.411	0.409	0.413	0.422	0.435	0.447
cdlib_dcs	0.416	0.401	0.391	0.397	0.399	0.405
cdlib_combo	0.394	0.385	0.385	0.389	0.394	0.403
louvain	0.394	0.386	0.385	0.390	0.394	0.403
walktrap	0.399	0.387	0.386	0.391	0.395	0.402
greedy_modularity	0.392	0.385	0.384	0.390	0.393	0.402
cdlib_der	0.716	0.700	0.696	0.694	0.691	0.693
cdlib_scan	0.413	0.399	0.394	0.396	0.397	0.403
affinity_propagation	0.709	0.698	0.695	0.693	0.691	0.693
ensemble_no_weight_0.5	0.404	0.390	0.388	0.391	0.396	0.402
ensemble_no_weight_0.6	0.394	0.388	0.384	0.389	0.395	0.402
ensemble_no_weight_0.7	0.383	0.380	0.381	0.389	0.393	0.403

Table 37: Results of the VoI metric of the clustering algorithms for each threshold on the full data of the Musicbrainz data set.

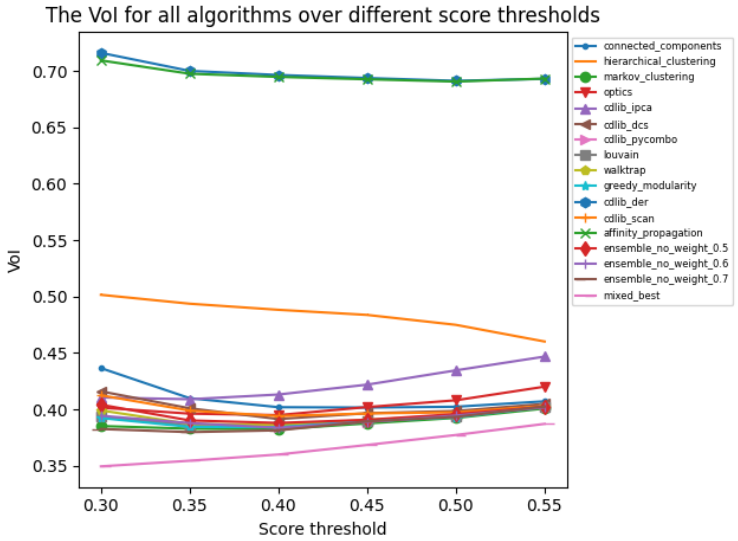

Figure 27: All algorithms on the Musicbrainz data set.

Musicbrainz - Test split

Algorithm	0.3	0.35	0.4	0.45	0.5	0.55
test_split_mixed_best	0.977	0.980	0.975	0.978	0.979	0.984
test_split_connected_components	0.816	0.860	0.758	0.886	0.925	0.938
test_split_hierarchical_clustering	0.747	0.793	0.747	0.798	0.835	0.872
test_split_markov_clustering	0.936	0.937	0.928	0.944	0.953	0.962
test_split_optics	0.911	0.931	0.934	0.940	0.938	0.945
test_split_ipca	0.901	0.907	0.905	0.905	0.896	0.903
test_split_dcs	0.894	0.899	0.904	0.922	0.941	0.956
test_split_combo	0.933	0.940	0.921	0.948	0.957	0.966
test_split_louvain	0.933	0.938	0.922	0.948	0.956	0.966
test_split_walktrap	0.921	0.936	0.905	0.941	0.953	0.965
test_split_greedy_modularity	0.933	0.940	0.922	0.947	0.957	0.964
test_split_der	0.412	0.408	0.368	0.411	0.431	0.423
test_split_scan	0.888	0.904	0.849	0.924	0.946	0.958
test_split_affinity_propagation	0.453	0.454	0.416	0.460	0.469	0.465
test_split_ensemble_no_weight_0.5	0.916	0.928	0.896	0.938	0.946	0.959
test_split_ensemble_weighted_0.5	0.755	0.795	0.622	0.623	0.523	0.852
test_split_ensemble_no_weight_0.6	0.925	0.933	0.919	0.944	0.953	0.964
test_split_ensemble_weighted_0.6	0.742	0.807	0.592	0.598	0.500	0.848
test_split_ensemble_no_weight_0.7	0.944	0.948	0.942	0.948	0.957	0.966
test_split_ensemble_weighted_0.7	0.741	0.805	0.580	0.602	0.498	0.847
test_split_predicted_clustering	0.860	0.891	0.851	0.880	0.850	0.825

Table 38: Results of the F1 metric of the clustering algorithms for each threshold on the test split of the Musicbrainz data set.

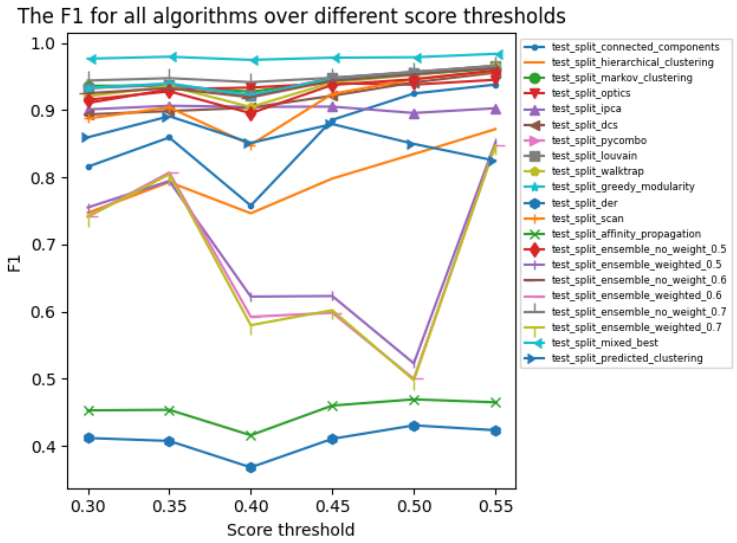

Figure 28: All algorithms on the Musicbrainz data set test split.

Algorithm	0.3	0.35	0.4	0.45	0.5	0.55
test_split_mixed_best	0.959	0.964	0.958	0.961	0.962	0.971
test_split_connected_components	0.697	0.774	0.640	0.804	0.862	0.885
test_split_hierarchical_clustering	0.838	0.904	0.781	0.842	0.875	0.885
test_split_markov_clustering	0.893	0.891	0.871	0.899	0.915	0.930
test_split_optics	0.872	0.905	0.915	0.925	0.933	0.945
test_split_ipca	0.923	0.933	0.942	0.950	0.953	0.967
test_split_dcs	0.812	0.826	0.829	0.858	0.892	0.919
test_split_combo	0.885	0.900	0.871	0.919	0.937	0.953
test_split_louvain	0.885	0.897	0.874	0.919	0.936	0.953
test_split_walktrap	0.861	0.889	0.837	0.898	0.925	0.944
test_split_greedy_modularity	0.886	0.899	0.873	0.917	0.937	0.948
test_split_der	0.645	0.726	0.562	0.752	0.831	0.853
test_split_scan	0.804	0.837	0.753	0.866	0.903	0.925
test_split_affinity_propagation	0.692	0.747	0.622	0.796	0.856	0.878
test_split_ensemble_no_weight_0.5	0.848	0.871	0.817	0.887	0.902	0.924
test_split_ensemble_weighted_0.5	0.755	0.824	0.713	0.742	0.791	0.915
test_split_ensemble_no_weight_0.6	0.866	0.881	0.858	0.900	0.920	0.936
test_split_ensemble_weighted_0.6	0.778	0.867	0.730	0.727	0.793	0.915
test_split_ensemble_no_weight_0.7	0.910	0.918	0.912	0.922	0.942	0.955
test_split_ensemble_weighted_0.7	0.807	0.879	0.744	0.745	0.798	0.917
test_split_predicted_clustering	0.899	0.919	0.865	0.935	0.922	0.939

Table 39: Results of the precision metric of the clustering algorithms for each threshold on the test split of the Musicbrainz data set.

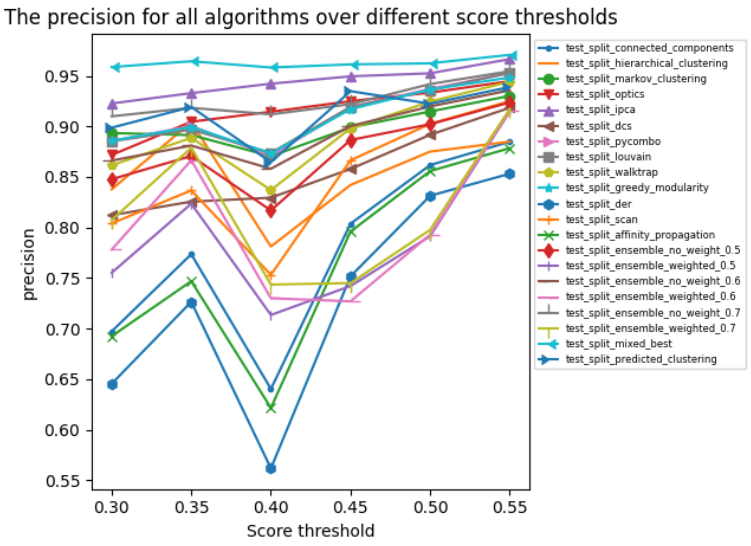

Figure 29: All algorithms on the Musicbrainz data set test split.

Algorithm	0.3	0.35	0.4	0.45	0.5	0.55
test_split_mixed_best	0.996	0.996	0.993	0.996	0.996	0.998
hline test_split_connected_components	1.000	1.000	1.000	1.000	1.000	1.000
test_split_hierarchical_clustering	0.677	0.707	0.730	0.767	0.800	0.861
test_split_markov_clustering	0.984	0.988	0.994	0.995	0.995	0.996
test_split_optics	0.954	0.960	0.954	0.956	0.943	0.946
test_split_ipca	0.881	0.882	0.871	0.865	0.846	0.847
test_split_dcs	0.995	0.998	0.995	0.998	0.997	0.997
test_split_combo	0.986	0.984	0.982	0.979	0.978	0.980
test_split_louvain	0.987	0.985	0.980	0.979	0.978	0.980
test_split_walktrap	0.990	0.989	0.990	0.990	0.983	0.987
test_split_greedy_modularity	0.987	0.986	0.981	0.980	0.979	0.980
test_split_der	0.309	0.295	0.299	0.288	0.291	0.282
test_split_scan	0.995	0.996	0.993	0.993	0.995	0.994
test_split_affinity_propagation	0.342	0.336	0.333	0.326	0.324	0.317
test_split_ensemble_no_weight_0.5	0.996	0.996	0.997	0.996	0.995	0.998
test_split_ensemble_weighted_0.5	0.776	0.808	0.612	0.590	0.448	0.851
test_split_ensemble_no_weight_0.6	0.994	0.993	0.993	0.993	0.990	0.993
test_split_ensemble_weighted_0.6	0.726	0.768	0.533	0.557	0.429	0.846
test_split_ensemble_no_weight_0.7	0.981	0.980	0.975	0.977	0.973	0.977
test_split_ensemble_weighted_0.7	0.702	0.748	0.511	0.553	0.427	0.841
test_split_predicted_clustering	0.825	0.865	0.853	0.831	0.789	0.737

Table 40: Results of the recall metric of the clustering algorithms for each threshold on the test split of the Musicbrainz data set.

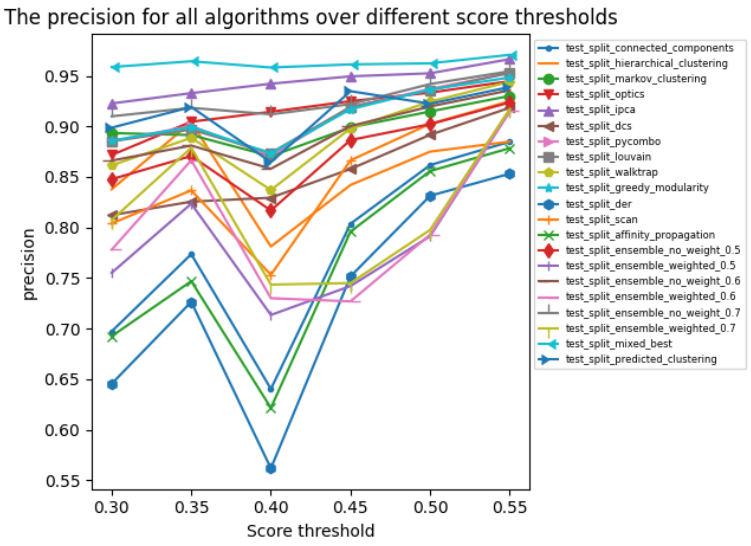

Figure 30: All algorithms on the Musicbrainz data set test split.

Algorithm	0.3	0.35	0.4	0.45	0.5	0.55
test_split_mixed_best	28.714	25.571	35.571	25.714	23.714	17.714
test_split_connected_components	108.857	82.857	75.429	58.000	48.714	39.143
test_split_hierarchical_clustering	355.714	309.000	295.429	256.000	220.571	163.286
test_split_markov_clustering	81.714	72.571	63.429	54.143	47.571	37.857
test_split_optics	138.143	110.714	104.857	90.857	87.571	75.143
test_split_ipca	148.000	133.857	138.714	130.429	139.429	125.714
test_split_dcs	109.571	83.000	78.429	60.286	51.143	41.000
test_split_combo	96.429	81.429	75.857	65.143	56.143	45.571
test_split_louvain	97.286	81.143	78.000	65.857	56.714	46.143
test_split_walktrap	97.000	79.571	73.000	59.714	54.143	43.429
test_split_greedy_modularity	97.286	79.857	76.857	64.571	55.143	46.857
test_split_der	779.286	765.000	765.571	742.143	720.143	701.857
test_split_scan	103.143	81.571	80.000	62.857	51.143	42.000
test_split_affinity_propagation	814.857	796.429	800.429	770.571	745.429	722.714
test_split_ensemble_no_weight_0.5	95.143	75.286	70.571	56.857	49.429	38.143
test_split_ensemble_weighted_0.5	300.143	238.429	469.286	488.286	613.143	178.571
test_split_ensemble_no_weight_0.6	92.429	75.429	69.429	55.857	50.286	39.286
test_split_ensemble_weighted_0.6	332.143	268.714	575.000	513.857	628.571	181.571
test_split_ensemble_no_weight_0.7	89.857	80.286	74.000	65.286	59.143	47.429
test_split_ensemble_weighted_0.7	348.429	287.857	594.714	518.286	630.571	184.143
test_split_predicted_clustering	265.857	221.143	235.143	234.429	274.571	284.571

Table 41: Results of the GMD metric of the clustering algorithms for each threshold on the test split of the Musicbrainz data set.

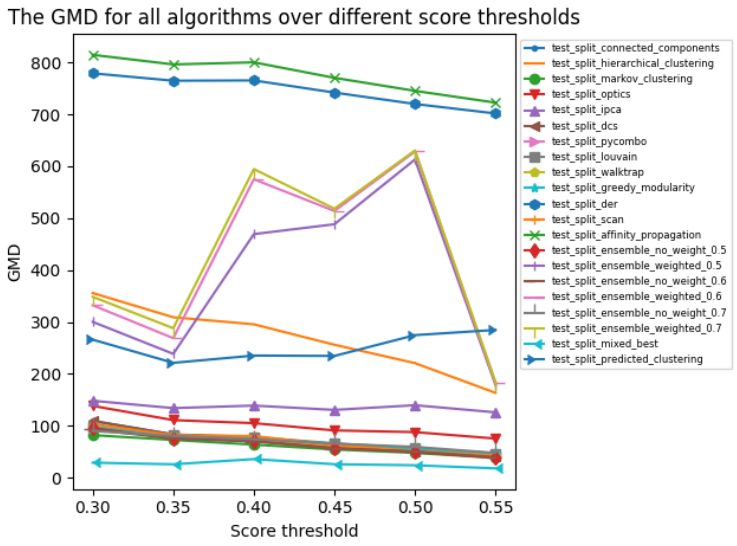

Figure 31: All algorithms on the Musicbrainz data set test split.

Algorithm	0.3	0.35	0.4	0.45	0.5	0.55
test_split_mixed_best	0.016	0.014	0.018	0.014	0.013	0.010
test_split_connected_components	0.083	0.059	0.069	0.044	0.035	0.027
test_split_hierarchical_clustering	0.174	0.148	0.148	0.127	0.109	0.080
test_split_markov_clustering	0.045	0.041	0.039	0.033	0.028	0.022
test_split_optics	0.070	0.055	0.052	0.045	0.045	0.038
test_split_ipca	0.075	0.069	0.070	0.067	0.072	0.065
test_split_dcs	0.067	0.053	0.052	0.040	0.033	0.025
test_split_combo	0.051	0.043	0.045	0.035	0.030	0.023
test_split_louvain	0.051	0.043	0.045	0.035	0.030	0.024
test_split_walktrap	0.055	0.044	0.047	0.035	0.030	0.023
test_split_greedy_modularity	0.051	0.042	0.045	0.035	0.029	0.024
test_split_der	0.365	0.353	0.362	0.338	0.325	0.314
test_split_scan	0.065	0.050	0.058	0.040	0.031	0.024
test_split_affinity_propagation	0.363	0.351	0.357	0.335	0.323	0.311
test_split_ensemble_no_weight_0.5	0.056	0.044	0.049	0.035	0.030	0.023
test_split_ensemble_weighted_0.5	0.154	0.123	0.217	0.219	0.271	0.081
test_split_ensemble_no_weight_0.6	0.052	0.043	0.044	0.033	0.029	0.022
test_split_ensemble_weighted_0.6	0.167	0.132	0.255	0.231	0.278	0.082
test_split_ensemble_no_weight_0.7	0.045	0.040	0.039	0.035	0.030	0.024
test_split_ensemble_weighted_0.7	0.173	0.139	0.263	0.232	0.278	0.083
test_split_predicted_clustering	0.120	0.097	0.107	0.101	0.120	0.127

Table 42: Results of the VoI metric of the clustering algorithms for each threshold on the test split of the Musicbrainz data set.

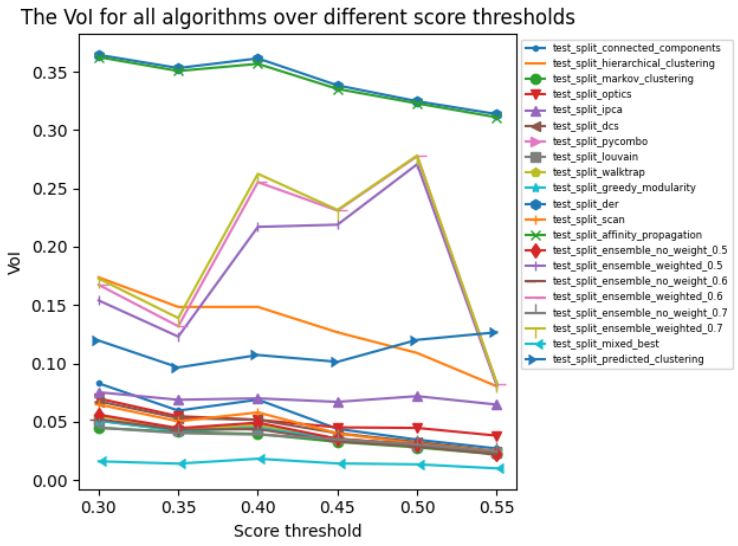

Figure 32: All algorithms on the Musicbrainz data set test split.

B Appendix - Ethics and Privacy Quick Scan

Response Summary:

Section 1. Research projects involving human participants

P1. Does your project involve human participants? This includes for example use of observation, (online) surveys, interviews, tests, focus groups, and workshops where human participants provide information or data to inform the research. If you are only using existing data sets or publicly available data (e.g. from Twitter, Reddit) without directly recruiting participants, please answer no.

- No

Section 2. Data protection, handling, and storage

The General Data Protection Regulation imposes several obligations for the use of **personal data** (defined as any information relating to an identified or identifiable living person) or including the use of personal data in research.

D1. Are you gathering or using personal data (defined as any information relating to an identified or identifiable living person)?

- Yes

High-risk data

DR1. Will you process personal data that would jeopardize the physical health or safety of individuals in the event of a personal data breach?

- No

DR2. Will you combine, compare, or match personal data obtained from multiple sources, in a way that exceeds the reasonable expectations of the people whose data it is?

- No

DR3. Will you use any personal data of children or vulnerable individuals for marketing, profiling, automated decision-making, or to offer online services to them?

- No

DR4. Will you profile individuals on a large scale?

- No

DR5. Will you systematically monitor individuals in a publicly accessible area on a large scale (or use the data of such monitoring)?

- No

DR6. Will you use special category personal data, criminal offense personal data, or other sensitive personal data on a large scale?

- No

DR7. Will you determine an individual's access to a product, service, opportunity, or benefit based on an automated decision or special category personal data?

- No

DR8. Will you systematically and extensively monitor or profile individuals, with significant effects on them?

- No

DR9. Will you use innovative technology to process sensitive personal data?

- No

Data minimization

DM1. Will you collect only personal data that is strictly necessary for the research?

- Yes

DM4. Will you anonymize the data wherever possible?

- Yes

DM5. Will you pseudonymize the data if you are not able to anonymize it, replacing personal details with an identifier, and keeping the key separate from the data set?

- Not applicable

Using collaborators or contractors that process personal data securely

DC1. Will any organization external to Utrecht University be involved in processing personal data (e.g. for transcription, data analysis, data storage)?

- Yes

DC2. Will this involve data that is not anonymized?

- No

International personal data transfers

DI1. Will any personal data be transferred to another country (including to research collaborators in a joint project)?

- No

Fair use of personal data to recruit participants

DF1. Is personal data used to recruit participants?

- No

Participants' data rights and privacy information

DP1. Will participants be provided with privacy information? (Recommended is to use as part of the information sheet: For details of our legal basis for using personal data and the rights you have over your data please see the University's privacy information at www.uu.nl/en/organisation/privacy.)

- Yes

DP2. Will participants be aware of what their data is being used for?

- Yes

DP3. Can participants request that their personal data be deleted?

- Yes

DP4. Can participants request that their personal data be rectified (in case it is incorrect)?

- Yes

DP5. Can participants request access to their personal data?

- Yes

DP6. Can participants request that personal data processing is restricted?

- Yes

DP7. Will participants be subjected to automated decision-making based on their personal data with an impact on them beyond the research study to which they consented?

- No

DP8. Will participants be aware of how long their data is being kept for, who it is being shared with, and any safeguards that apply in case of international sharing?

- Yes

DP9. If data is provided by a third party, are people whose data is in the data set provided with (1) the privacy information and (2) what categories of data you will use?

- Yes

Using data that you have not gathered directly from participants

DE1. Will you use any personal data that you have not gathered directly from participants (such as data from an existing data set, data gathered for you by a third party, data scraped from the internet)?

- Yes

DE2. Will you use an existing dataset in your research?

- Yes

DE3. Do you have permission to do so from the owners of the data set?

- Yes

DE4. Have the people whose data is in the data set consented to their data being used by other researchers and/or for purposes other than that for which that data set was gathered?

- Yes

DE5. Are there any contractual conditions attached to working with or storing the data from DE2?

- No

DE6. Does your project require access to personal data about participants from other parties (e.g., teachers, employers), databanks, or files?

- No

DE9. Does the project involve collecting personal data from websites or social media (e.g., Facebook, Twitter, Reddit)?

- No

Secure data storage

DS1. Will any data be stored (temporarily or permanently) anywhere other than on password-protected University authorized computers or servers?

- No

DS4. Excluding (1) any international data transfers mentioned above and (2) any sharing of data with collaborators and contractors, will any personal data be stored, collected, or accessed from outside the EU?

- No

Section 3. Research that may cause harm

Research may cause harm to participants, researchers, the university, or society. This includes when technology has dual-use, and you investigate an innocent use, but your results could be used by others in a harmful way. If you are unsure regarding possible harm to the university or society, please discuss your concerns with the Research Support Office.

H1. Does your project give rise to a realistic risk to the national security of any country?

- No

H2. Does your project give rise to a realistic risk of aiding human rights abuses in any country?

- No

H3. Does your project (and its data) give rise to a realistic risk of damaging the University's reputation? (E.g., bad press coverage, public protest.)

- No

H4. Does your project (and in particular its data) give rise to an increased risk of attack (cyber- or otherwise) against the University? (E.g., from pressure groups.)

- No

H5. Is the data likely to contain material that is indecent, offensive, defamatory, threatening, discriminatory, or extremist?

- No

H6. Does your project give rise to a realistic risk of harm to the researchers?

- No

H7. Is there a realistic risk of any participant experiencing physical or psychological harm or discomfort?

- No

H8. Is there a realistic risk of any participant experiencing a detriment to their interests as a result of participation?

- No

H9. Is there a realistic risk of other types of negative externalities?

- No

Section 4. Conflicts of interest

C1. Is there any potential conflict of interest (e.g. between research funder and researchers or participants and researchers) that may potentially affect the research outcome or the dissemination of research findings?

- No

C2. Is there a direct hierarchical relationship between researchers and participants?

- No

Section 5. Your information.

This last section collects data about you and your project so that we can register that you completed the Ethics and Privacy Quick Scan, sent you (and your supervisor/course coordinator) a summary of what you filled out, and follow up where a fuller ethics review and/or privacy assessment is needed. For details of our legal basis for using personal data and the rights you have over your data please see the University's privacy information. Please see the guidance on the ICS Ethics and Privacy website on what happens on submission.

Z0. Which is your main department?

- Information and Computing Science

Z1. Your full name:

Frank Daniël Hoogmoed

Z2. Your email address:

f.d.hoogmoed@students.uu.nl

Z3. In what context will you conduct this research?

- As a student for my master thesis, supervised by::
  Dr. I.R. Ioana Karnstedt-Hulpus

Z5. Master programme for which you are doing the thesis

- Computing Science

Z6. Email of the course coordinator or supervisor (so that we can inform them that you filled this out and provide them with a summary):

i.r.karnstedt-hulpus@uu.nl

Z7. Email of the moderator (as provided by the coordinator of your thesis project):

i.velegrakis@uu.nl

Z8. Title of the research project/study for which you filled out this Quick Scan:

Data imputation for Entity resolution, with a focus on applying clustering methods with missing similarities in adjacency matrices and imputation methods for missing similarities in adjacency matrices

Z9. Summary of what you intend to investigate and how you will investigate this (200 words max):

The aim is to improve an existing entity resolution pipeline by investigating different methods for clustering and data imputation. All the while, keeping in consideration the fact that the pipeline is used for large datasets, meaning the methods have to also provide results in reasonable time.
If other findings improve the precision or the speed of entity resolution, research/implement those aswell.
Therefor, mostly literature and combinations of different researches will be explored to perform a survey on different techniques.

Z10. In case you encountered warnings in the survey, does supervisor already have ethical approval for a research line that fully covers your project?

- Not applicable

Scoring

- Privacy: 0
- Ethics: 0
